BETRIEBLICHE UMWELTDATENBERICHTERSTATTUNG NACH DEM P23R-PRINZIP – NUTZUNG STANDARDISIERTER BERICHTSKOMPONENTEN (P23R4FLEX)

Umwelt Bundesamt

Berichtskennblatt

Aktenzeichen: UBA II 2.2 20500-3/5	Vorhaben-Nr.:
Titel des Vorhabens: Betriebliche Umweltdatenberichterstattung nach dem P23R-Prinzip – Nutzung standardisierter Berichtskomponenten (P23R4FLEX)	Autor(en): Dr. Joachim Heidemeier, Ulrike Schüler, Susanne von Kurnatowski, Antje Ullrich
Vorhabensbeginn: 01.08.2012	Vorhabenende (Abschlussdatum): 31.05.2014
Veröffentlichungsdatum: 31.07.2014	Seitenzahl: 201
Fördernehmer/ -in (Name, Anschrift): Umweltbundesamt, FG II 2.2, Wörlitzer Platz 1, 06844 Dessau-Roßlau	
Gefördert im Rahmen der eGovernment-Aktivitäten des Bundesministeriums des Innern	
Kurzfassung/Summary: Das Projekt P23R4FLEX will zu einem automatisierten, optimierten und qualitätsgesicherten Datenaustausch zwischen den Beteiligten an solchen Berichtsprozessen (zwischen Wirtschaft und Verwaltung sowie zwischen unterschiedlichen Verwaltungen) beitragen. Es baut auf den Ergebnissen und Erfahrungen vorangegangener Projekte (P23R, XUBetrieb, MIFLEX) auf und steht für die Verknüpfung von flexiblen Komponenten (FLexible Data EXchange) in der Umweltberichterstattung mit Hilfe des P23R-Prinzips (Prozessdatenbeschleuniger). Ziel von P23R4FLEX ist, in der betrieblichen Umweltberichtserstattung modellhaft die Anwendbarkeit des P23R-Prinzips zu zeigen. Im Umweltbereich die Qualitätssicherung der zu berichtenden Daten als außerordentlich wichtig herausgestellt. Dieser Aspekt steht ebenfalls im Fokus des Projektes P23R4FLEX.	
Schlagwörter: P23R, Prozessdatenbeschleuniger, Umweltdatenberichterstattung, Berichtsprozess, Berichterstattung, eGovernment, Wasserhaushaltsgesetz, EU-Kommunalabwasserrichtlinie, PRTR	
Anzahl der gelieferten Berichte Papierform: 1 Elektronischer Datenträger: 1	Veröffentlichung im Internet auf der Homepage: www.p23r4flex.de

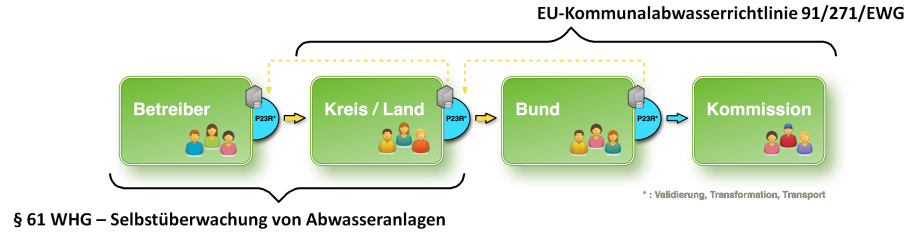
Inhaltsverzeichnis

Abbildungsverzeichnis

Kurzfassung

Informations- und Meldepflichten im Umweltbereich folgen national oder international rechtlich festgeschriebenen Anforderungen. Ihre Umsetzung erfordert oftmals mehrstufige Prozessketten, z.B. vom Betreiber über unterschiedliche Landes- und Bundesbehörden bis zur EU.
Das Projekt P23R4FLEX will zu einem automatisierten, optimierten und qualitätsgesicherten Datenaustausch zwischen den Beteiligten an solchen Berichtsprozessen (zw. Wirtschaft und Verwaltung sowie zwischen unterschiedlichen Verwaltungsbehörden) beitragen. Es baut auf den Ergebnissen und Erfahrungen vorangegangener Projekte (P23R, XUBetrieb, MIFLEX) auf und steht dabei für die Verknüpfung von flexiblen Komponenten (FLexible Data EXchange) in der Umweltberichterstattung mit Hilfe des P23R-Prinzips (Prozessdatenbeschleuniger). Ziel von P23R4FLEX ist, in der betrieblichen Umweltberichtserstattung modellhaft die Anwendbarkeit des P23R-Prinzips zu zeigen. Im Umweltbereich hat sich die Qualitätssicherung der zu berichtenden Daten als außerordentlich wichtig herausgestellt. Dieser Aspekt steht ebenfalls im Fokus des Projektes P23R4FLEX. Daher werden beispielhaft zwei Berichtsprozesse aus dem Umweltbereich betrachtet:

- EU-Kommunalabwasserrichtlinie (Art. 15(4) 91/271/EWG)
- Wasserhaushaltsgesetz (WHG) §61 - Selbstüberwachung bei Abwassereinleitungen und Abwasseranlagen



Beide Berichtsprozesse greifen sowohl hinsichtlich der zu übermittelnden Dateninhalte als auch der Prozessbeteiligten auf der Ebene der Bundesländer ineinander (s. Abb.). Ein großer Teil der Berichtsdaten der EU-Kommunalabwasser-Richtlinie werden über die Anforderungen des § 61 WHG durch die Betreiber den zuständigen Landesbehörden übermittelt. In der Berichterstattung zur EU-Kommunalabwasser-Richtlinie werden die Bundesländer gebeten, die Daten bereit zu stellen.
Das Ziel des Projektes - Vorbereitung einer breiten Einführung und Anwendung des Prozessdatenbeschleuniger-Prinzips (P23R-Prinzips) in der Kommunikation zwischen Anlagenbetreiber und Behörde - wurde mit dem beschriebenen Projekt erreicht. Folgende grundlegende Ergebnisse wurden im Projekt erreicht:

- Prozessbeschreibungen inkl. Qualitätsanforderungen, Qualitätsprüfungen
- Programmierung der notwendigen Schnittstellen

- Entwicklung zweier Regeln zum Prozess Kommunalabwasserrichtlinie und einer Regel zum Prozess §61 Wasserhaushaltsgesetz
- Konzept für die Architektur und Funktionalität eines P23R4FLEX-Clients
- Analyse von XML-Schemasprachen für die Qualitätssicherung im Kontext des P23R-Prinzips
- Entwurf Handbuch zur Regelerstellung

Der dem P23R zugrunde liegende Ansatz stellte sich als richtige, realisierbare und praxistaugliche Idee dar. Einige der im ursprünglichen P23R-Prinzip ausspezifizierten, technischen Randbedingungen für eine praxisorientierte Anwendung haben sich jedoch bei komplexeren Berichtspflichten nicht als die optimale Lösung darstellt. Diesbezüglich sollte die P23R-Spezifikation einem Review, unter Berücksichtigung der Praxiserfahrungen, unterzogen und ggf. angepasst werden. Die Musterimplementierung von Fraunhofer FOKUS ist momentan die einzige, von der Zielsetzung offene Implementierung wichtiger Funktionsblöcke des P23R-Konzeptes.
Vorteile für Betreiber und Behörden durch den Einsatz von P23R liegen u.a. in einer vereinfachten Datenhaltung und -pflege sowie in qualitativ hochwertigeren Daten, die (im Umweltbereich) in eine bessere (Umwelt)Politik münden können bei gleichzeitig geringeren Kosten für die Datenbereitstellung. Nicht immer sind diese Vorteile allen Beteiligten gleichermaßen ersichtlich und wichtig, sicherlich auch, weil bei allen Beteiligten Anfangsinvestitionen getätigt werden müssen. Wichtig ist weiterhin die Einstiegshürden für Anwender und Infrastrukturbereitsteller merklich zu reduzieren (u.a. durch weit verbreitete Werkzeuge und offene Lizenzen). Die Klärung der Lizensierungsfrage der P23R-Musterimplementierung ist eine Grundvoraussetzung, um eine offene Weiterentwicklung der Plattform zu ermöglichen. Voraussetzung für eine breite Einführung des P23R-Prinzips in die Praxis ist auch eine kontinuierliche und zielgerichtete Öffentlichkeitsarbeit.

1. Einleitung

Der Projektname P23R4FLEX steht für die Kombination aus Prozessdatenbeschleuniger und dem Claim „Flexible Environmental Data Exchange“ (FLEX). P23R4FLEX ist dem OpenSource-Gedanken verpflichtet und veröffentlicht alle Dokumentationen unter der Free Documentation License 1.3, Datenbanken unter der Open Database License und alle anderen Ergebnisse unter einer offenen Lizenz, die sich an der für die P23R-Musterimplementierung orientiert. Das Bundesministerium des Innern plant, letztere zukünftig unter eine offene Lizenz zu stellen, eine endgültige Entscheidung steht derzeit noch aus.

1.1. Ausgangssituation

Was ist der P23R?

Ziel des Prozess-Daten-Beschleunigers P23R (www.p23r.de) ist es, die Kommunikation zwischen Wirtschaft und Verwaltung sowie Verwaltungsverfahren zeitlich und organisatorisch zu optimieren und damit eine deutliche Senkung von Bürokratiekosten zu ermöglichen. Der P23R setzt dabei auf ein Infrastrukturkonzept, das die Datenhoheit bei den Unternehmen belässt und diese in die Lage versetzt, ihren gesetzlich vorgeschriebenen Berichtspflichten nachzukommen, wobei bestehende Lösungen ergänzt und erweitert werden können. Lösungen auf Basis des P23R-Prinzips generieren die erforderlichen Meldungen automatisch und stellen sie den zuständigen Behörden nach Prüfung ordnungsgemäß zu. Gleichzeitig sollen Unternehmen von Detailregelungen, Plausibilitätsprüfungen und Aktualisierungen entlastet werden. Behörden erhalten durch die Nutzung effizienterer, weil medienbruchfreier Prozessketten schneller Daten in deutlich höherer Qualität.
Das P23R-Prinzip wurde im Auftrag des Bundesministerium des Innern (BMI) formuliert. Im Rahmen eines Projektes (http://www.p23r.de/projekttinformationen/) in den Jahren 2010/ 2011 wurde ein P23R-Prototyp entwickelt und die Machbarkeit gezeigt. Testdomänen für die Entwicklung von P23R waren die Bereiche Arbeitgebermeldepflichten und Umweltberichterstattung. Die Ergebnisse und die Pilotanwendungen wurden in der Metropolregion Rhein-Neckar getestet.

Rolle des Umweltbundesamtes

Das Umweltbundesamt (UBA) ist als Bundesoberbehörde im Geschäftsbereich des BMUB zentrale Kontaktstelle für zahlreiche Berichtspflichten der Bundesrepublik an supranationale Organisationen, insbesondere an die Europäische Union. Diese Berichtspflichten beruhen in der Regel auf komplexen, mehrstufigen Prozessketten unter Einbeziehung von Betreibern, ggf. externen Sachverständigen, Kommunal- und Landesbehörden. Die Berichterstellungen sind bislang häufig durch fremdbestimmte, kurzfristige, ad-hoc definierte Prozesse mit fachlich überlappenden, aber technisch

unterschiedlich spezifizierten Berichtsinhalten gekennzeichnet. Das UBA bemüht sich aktiv darum, bei der Gestaltung von Berichtsprozessen Bürokratieaufwände zu vermindern. So konsolidiert die Berichterstattung zum deutschen Schadstofffreisetzungs- und -verbringungsregister PRTR unter Nutzung von BUBE-online gleichzeitig die Prozessketten zweier weiterer Berichtspflichten: der 11. BImSchV (Bundesimmissionsschutzverordnung über Emissionserklärungen) und der 13. BImSchV (über Großfeuerungs-, Gasturbinen und Verbrennungsmotoranlagen). Aufgrund seiner Erfahrungen in der Umweltberichterstattung hat das Umweltbundesamt in den Jahren 2010-2011 das Projekt "XUBetrieb" zur Standardisierung von Berichtsinhalten in der betrieblichen Umweltberichterstattung durchgeführt. Darin wurden, basierend auf dem XÖV-Leitfaden und einer detaillierten Prozessanalyse von insgesamt 13 besonders aufwendigen Berichtspflichten, eine Komponentenbibliothek für betriebliche Stamm- und Berichtsdaten im Umweltbereich entwickelt. Mit Hilfe dieser Komponentenbibliothek können Datenmodelle für den Aufbau von Nachrichten und Datenbanken nun auf einer standardisierten Basis zusammengestellt werden, indem die passenden Bausteine für eine Aufgabe selektiert und zusammengefügt werden. Im Rahmen des Projektes wurde dies zusätzlich anhand des Datenmodells für die Berichterstattung nach EU-Kommunalabwasserrichtlinie an der Schnittstelle Land-Bund demonstriert. Neben der XöV-Konformität sind die Bibliotheks-Komponenten auch UN/CEFACT-konform und gleichzeitig mit englischen Bezeichnern versehen, so dass auch der internationale Einsatz ermöglicht wird. Die XUBetrieb-Komponenbibliothek in der Version 1.0.0 erhielt im April 2012 die XöV-Zertifizierung, ebenso wie die aus XUBetrieb-Komponenten zusammengesetzte Schnittstelle XUKommunalabwasser.

Hindernisse in der derzeitigen Datenberichterstattung

Der bisher praktizierte Datenaustausch ist z.T. unsicher und kompliziert. Bedingt dadurch, dass die Qualitätssicherung der Daten oft nicht am Anfang der Prozesskette steht, wegen Medienbrüchen und weil Daten z.T. noch per Hand zusammengesucht werden, werden keine qualitätsgesicherten und z.T. unplausible Daten weitergereicht. Dies führt zu einem hohen Bearbeitungsaufwand bei Betreibern und Behörden, wenn die Qualitätssicherung erst später stattfindet und dann korrigiert werden muss. Gerade bei Medienbrüchen kommt es oft zu Übertragungsfehlern. Daher wurde auf die Medienbruchfreiheit im vorliegenden Projekt viel Wert gelegt.
Derzeit ist die Datenhaltung in den Unternehmen und bei Behörden oftmals recht kompliziert, d.h. u.U. wird für jede Berichtspflicht eine separate Datenbank/Datenhaltung und ein anderes System betrieben, obwohl in verschiedenen Berichtspflichten teilweise identische Daten benötigt werden (oder zu berichtende Daten aus identischen Grundlagendaten zusammengefasst/ abgeleitet werden). Müssen Daten im Notfall korrigiert werden (Adresse o.ä.), dann ist diese Änderung bislang immer in allen Systemen notwendig.
Weitere Probleme treten bei unstrukturierten Anforderungen auf (Freitext, nicht automatisch befüllbar, nicht automatisch auswertbar). Oftmals steht bei der Definition der Anforderungen an eine Datenerhebung auch nicht im Vordergrund, wofür die erhobenen Daten letztlich benötigt werden (was will man daraus ableiten, welchen Auswertungen sollen sie dienen). Ideal wäre, wenn der Adressat einer Berichtspflicht (bspw. EU-Kommission bei europäischen Berichtspflichten, Bundesministerien bei na-

tionalen Berichtspflichten etc.) vor dem Aufsetzen derselben genau definierte, welchem Zweck sie dienen soll, was aus den Daten abgeleitet und welche Fragen beantwortet werden sollen, daraus dann Indikatoren abzuleiten, die dazu dienten festzulegen, welche Daten in welcher Granularität wie erhoben werden sollen - und dies in strukturierte, elektronische Anforderungen umzusetzen.
Die Berichterstattung mittels P23R soll viele der genannten Probleme beheben oder zumindest lindern. Durch den automatisierten, optimierten und qualitätsgesicherten Datenaustausch sowie flexible, qualitätsgesicherte und automatisierte Berichtsprozesse entstehen umfangreiche Vorteile:

- schnellere Prozessbearbeitung
- integrierte, automatisierte QS
- entlastete Fachabteilungen
- vereinfachte Datenverwaltung
- Transformations- und Übertragungsfehler vermeiden
- erleichterte Akzeptanz der Öffentlichkeit durch mehr Transparenz.

1.2. Zielstellung

Das Ziel des vorliegenden Projektes war die Vorbereitung einer breiten Einführung und Anwendung des Prozessdatenbeschleuniger-Prinzips (P23R-Prinzips) in der Kommunikation zwischen Anlagenbetreiber und Behörden und zwischen Behörden. Dazu wurde im Projekt P23R4FLEX die Idee des P23R (die das „wie" der Berichterstattung betrifft) mit den Ergebnissen des Projekts XUBetrieb (die das „was" der Berichterstattung betreffen) zusammengeführt und erstmals P23R-Regeln für die Berichterstattung entwickelt.
Die Prozeßdomäne „betriebliche Umweltdatenberichterstattung" bot sich für eine modellhafte Praxiserprobung und Weiterentwicklung mit dem Ziel einer Einführung des P23R-Prinzips an. Die Sicherstellung der Übertragbarkeit der gefundenen Lösungen auf andere Berichtsprozesse und –domämen war ein wichtiges Untersuchungsziel zur Verbesserung der E-Governmentangebote. Aufgrund seiner zentralen Rolle in zahlreichen Umwelt-Berichtsprozessen ist das Umweltbundesamt ein wichtiger Partner in diesem Bereich, insbesondere da auf langjährige bestehende Netzwerke sowohl mit europäischen Institutionen, den Bundesländern als auch mit der Industrie aufgebaut werden konnte.
Im Projekt wurden die P23R-Musterimplementierung auf die im Rahmen des XUBetrieb-Projektes analysierten Berichtsprozesse angewendet und durchgängig XUBetrieb-konforme Nachrichten für die Kommunikation verwendet. Damit sind beiden Bausteine für einen optimierten Datenaustausch zwischen Wirtschaft und Verwaltung, das "wie" und das "was" der Berichterstattung, miteinander verknüpft. Eine Integration über mehrere Verantwortlichkeitsstufen, wie sie beispielsweise für die Berichte an die EU erforderlich sind, ist mit dem P23R-Konzept realisierbar.
Ein weiterer Aspekt, der vertiefter Untersuchung und Erfahrung bedurfte, war die Integration der Qualitätssicherung in den Gesamtprozess. Als entscheidend für die tatsächliche Aufwandsreduzierung im Umweltbereich hat sich in der Praxis nämlich die

Anwendung eines umfassenden Qualitätssicherungskonzeptes herausgestellt, welches nicht nur formale Korrektheit sicherstellt, sondern auch semantische und fachliche Prüfungen so früh wie möglich in die Prozesskette integriert. Es fehlten zu Projektbeginn praktische Erfahrungen, inwieweit sich diese Anforderungen, die weit in den Bereich fachlicher Plausibilitätstest reichen, im Rahmen der aktuellen Version des P23R abbilden ließen, oder ob hier entsprechende Erweiterungen notwendig würden. Auch hier war die Übertragbarkeit in andere Berichtsdomänen ein wichtiger Untersuchungsgegenstand. Daher wurden die im Rahmen des XUBetrieb-Projektes untersuchten Berichtsprozesse unter Anwendung des P23R-Prinzips nachgebildet und getestet.
Durch die Arbeiten wurden neben einer Implementierung des P23R-Prinzips in zwei arbeitsaufwendigen Berichtspflichten im Umweltbereich gleichzeitig wichtige Vorarbeiten für die Umsetzung weiterer Berichtspflichten sowie der IE-Richtlinie geleistet.

1.3. Auftragnehmer

In einer ersten Ausschreibungsphase wurden die Arbeitspakete 2 und 3 und Teile des Arbeitspaketes (AP) 1 (s. Abb. 2.1) mit einer Laufzeit bis zum 30.04.2013 im Rahmen einer öffentlichen Ausschreibung an die Bietergemeinschaft „XUP23R“ als einzigem Bieter vergeben. Die Bietergemeinschaft „XUP23R“ bestand aus den Mitgliedern ENDA KG, Berlin und und binfort GmbH, Berlin. Es sollten zuerst Prozess- und Nachrichtenmodellen erstellt und dokumentiert werden. Die ENDA KG verfügt über umfangreiche Erfahrungen in der deutschen und europäischen Umweltberichterstattung und besitzt spezielle Kenntnisse der damit verbundenen Prozessketten. Die binfort GmbH brachte als Schwerpunkt nicht nur die Kenntnisse und Anwendung mehrerer Modellierungsnotationen für Geschäftsprozesse ein, sondern sie war darüber hinaus an der Entwicklung der Unified Modelling Methodology Version 2 der UN/CEFACT beteiligt.
In zwei weiteren Ausschreibungen wurden im Sommer 2013 die AP 4-11 (s. Abb. 2.1) an Fraunhofer FOKUS (FhG; mit Subauftragnehmer Atos IT Solution GmbH) und ENDA KG vergeben. Zwei Prozesse aus dem abwasserrechtlichen Vollzug sollten exemplarisch als wirkbetriebsfähige Prozesse testweise realisiert werden EU-Kommunalabwasserrichtlinie (Art. 15(4) 91/271/EWG), §61 Wasserhaushaltsgesetzes (WHG).
Sowohl FhG als auch Atos waren zusammen mit 11 weiteren Institutionen aus Wirtschaft und Wissenschaft Projektpartner im vom BMI finanzierten Projekt zur Entwicklung des P23R-Prinzips bis Ende 2011. FhG hat maßgeblich an der Entwicklung der Gesamtkonzeption des P23R-Infrastrukturprinzips mitgewirkt, die T-BRS Regelsprache spezifiziert und die Musterimplementierung, die in der Fachdomäne Umwelt bei der BASF genutzt wurde, entwickelt. Atos hat maßgeblich die Pilotierung in der Fachdomäne Umwelt bei der BASF durchgeführt und damit die einzigen vorliegenden T-BRS-Regeln im Umweltbereich geschrieben. Damit verfügten ausschließlich FhG und Atos über das notwendige IT-Architekturwissen sowie die zwingend notwendigen Erfahrungen bei der Regelerstellung.

1.4. Untersuchungsprozesse

Im Projekt P23R4FLEX wurden folgende zwei Berichtsprozesse aus dem Umweltbereich betrachtet:

- EU-Kommunalabwasserrichtlinie (Art. 15(4) 91/271/EWG)
- Wasserhaushaltsgesetz (WHG) - §61 Selbstüberwachung bei Abwassereinleitungen und Abwasseranlagen

Die beiden Prozesse wurden ausgewählt, da sie datenmäßig ineinander greifen und das UBA die Abwicklung der Kommunalabwasserrichtlinie bereits das 4. Mal durchführt und damit den Prozess gut kennt. Besonders die Berichterstattung nach der Kommunalabwasserrichtlinie ist ein idealer Kandidat für P23R. Er zeichnet sich durch folgende Aspekte aus: realistische Frequenz, leistbarer Zeitrahmen, strukturiertes, recht komplexes Datenmodell (m:n Relationen), Compliance Indikatoren / festgelegte Regeln, Object Lifetime Management Regeln und er ist für automatisierbare Screeningauswertungen geeignet.
Bei der Selbstüberwachung von Abwassereinleitungen und Abwasseranlagen handelt es sich um einen rein auf Landesebene ablaufenden Berichtsprozess, so dass das UBA nicht tangiert ist. Die von den Betreibern berichteten Daten stellen jedoch (von Genehmigungsdaten von Behördenseite) einen wichtigen Teil der dann über die Kommunalabwasserrichtlinie bis an die EU berichteten Daten dar. Die Abbildung zeigt, dass durch dieses Ineinandergreifen zweier Berichtspflichten die gesamte Prozesskette vom Betreiber über das Land zum Bund und schließlich zur EU mit dem P23R abgedeckt wird.

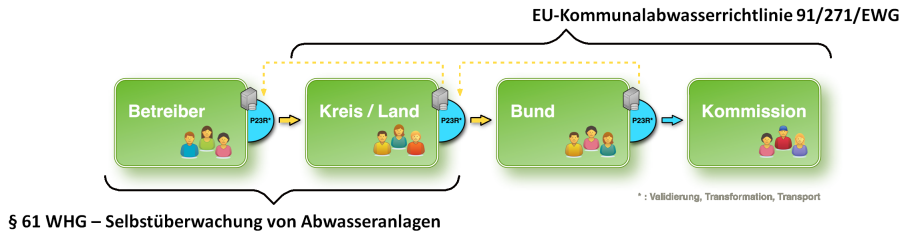


Abbildung 1.1.: Prozessketten im Überblick

EU-KommunalabwasserRL (Art.15(4) 91/271/EWG)

Im Rahmen der Berichterstattung zur EU-Kommunalabwasserrichtlinie sind die Mitgliedstaaten verpflichtet, auf Anfrage der EU-Kommission regelmäßig (aktuell im 2-Jahreszyklus) Daten zur Abwassersituation zu berichten. Diese Daten umfassen Informationen zu der in Haushalten und Gewerbe anfallenden Abwasserlast sowie zu kommunalen Kläranlagen und deren Einleitungen in die Gewässer, allerdings nur in Siedlungsgebieten ab einer Größe von 2.000 Einwohnerwerten (EW). Deutschland berichtete in den vorangegangenen Berichtsjahren Informationen zu jeweils ca. 4.300 Siedlungsgebieten und kommunalen Kläranlagen an die EU. Auswertungen aus diesen Daten werden ebenso regelmäßig von der EU-Kommission im Internet als Geodaten im WISE (Water Information System of Europe) veröffentlicht. Sie unterliegen daher den einschlägigen Anforderungen der INSPIRE-Richtlinie.

Auslöser eines europaweiten Berichtsprozesses ist die EU-Kommission. In Deutschland werden nach Eintreffen der EU-Anfrage über das BMU und das UBA die Ministerien der Bundesländer informiert und gebeten, die notwendigen Berichtsdaten in einem festgelegten Zeitraum dem UBA zur Verfügung zu stellen. Am UBA laufen (derzeit über ein zentrale Webanwendung e-Kommunalabwasser) die Daten aus den 16 Bundesländern in einer zentralen Datenbank zusammen. In der Webanwendung sind dabei eine Vielzahl von Qualitätssicherungsprüfungen automatisiert hinterlegt, welche die Daten auf Vollständigkeit, Einhaltung der Formate und Plausibilität prüfen. Bundeslandspezifisch liegen die Daten bei den zuständigen Landesumweltämtern entweder zentral vor oder müssen bei untergeordneten Behörden auf Landes- Kreis- oder regionaler Ebene angefragt werden.

§ 61 Selbstüberwachung bei Abwassereinleitungen und Abwasseranlagen

Im § 61 des Wasserhaushaltsgesetzes ist die Selbstüberwachung von Abwassereinleitungen und Abwasseranlagen geregelt. Dabei sind Anlagenbetreiber verpflichtet, den Zustand, die Funktionsfähigkeit, die Unterhaltung und den Betrieb der Abwasseranlage zuzüglich Art und Menge des Abwassers und der Abwasserinhaltsstoffe zu überprüfen. Hierüber sind Aufzeichnungen anzufertigen und auf Nachfrage der zuständigen Behörde im Bundesland zu übermitteln. Die Umsetzung der Berichterstattung zum § 61 WHG, d.h. sowohl Art und Umfang der Datenübermittlung als auch die Zuständigkeiten, liegt in der Hoheit der Bundesländer und wird in der jeweiligen Landesgesetzgebung von Bundesland zu Bundesland sehr unterschiedlich festgelegt.

2. Vorgehen

Der Zeitplan der umfangreichen Arbeiten im Projekt war durch die kurze Laufzeit straff organisiert. In den ersten Monaten erfolgte die Herstellung der vollen Kompatibilität zw. dem P23R-Datenmodell für Umwelt und dem XUBetrieb-Standard. Dieser wurde in einem Vorgängerprojekt (http://www.xubetrieb.de/) am UBA entwickelt und als Grundlage verwendet. Danach erfolgte die Optimierung des im XUBetrieb-Projektes analysierten Prozesses XUKommunalabwasser. Hierzu wurden Testfälle, -daten, -nachrichten und -benachrichtigungen entwickelt. Als Ergebnis liegt ein aufbereiteter XUKommunalabwasser-Prozess vor.
Im weiteren wurde XUKommunalabwasser P23R-konform umgesetzt und gleichzeitig der Prozess §61 WHG analog zum oben beschriebenen Vorgehen aufbereitet. Nach der P23R-konformen Umsetzung von XUKommunalabwasser erfolgte dessen Test sowie die Entwicklung zum wirkbetriebsfähigen Prozess. Danach wurde der §61 WHG ebenfalls P23R-konform umgesetzt und getestet. Um später weitere Prozesse schnell und einfach P23R-konform umsetzten zu können, wurden im Anschluss wiederverwendbare Bausteine ermittelt und ein Handbuch zur Regelerstellung entworfen. Ein Konzept für die Architektur und Funktionalität eines P23R4FLEX Clients rundeten die Arbeiten ab.

2.1. Projektablauf

Die folgende Grafik stellt den Projektplan zu Beginn des Projektes dar. Während des Projektes kam es aufgrund von technischen Problemen immer wieder zu Verzögerungen. Zum Teil wurden andere Arbeiten vorgezogen, bis die Probleme behoben haben.

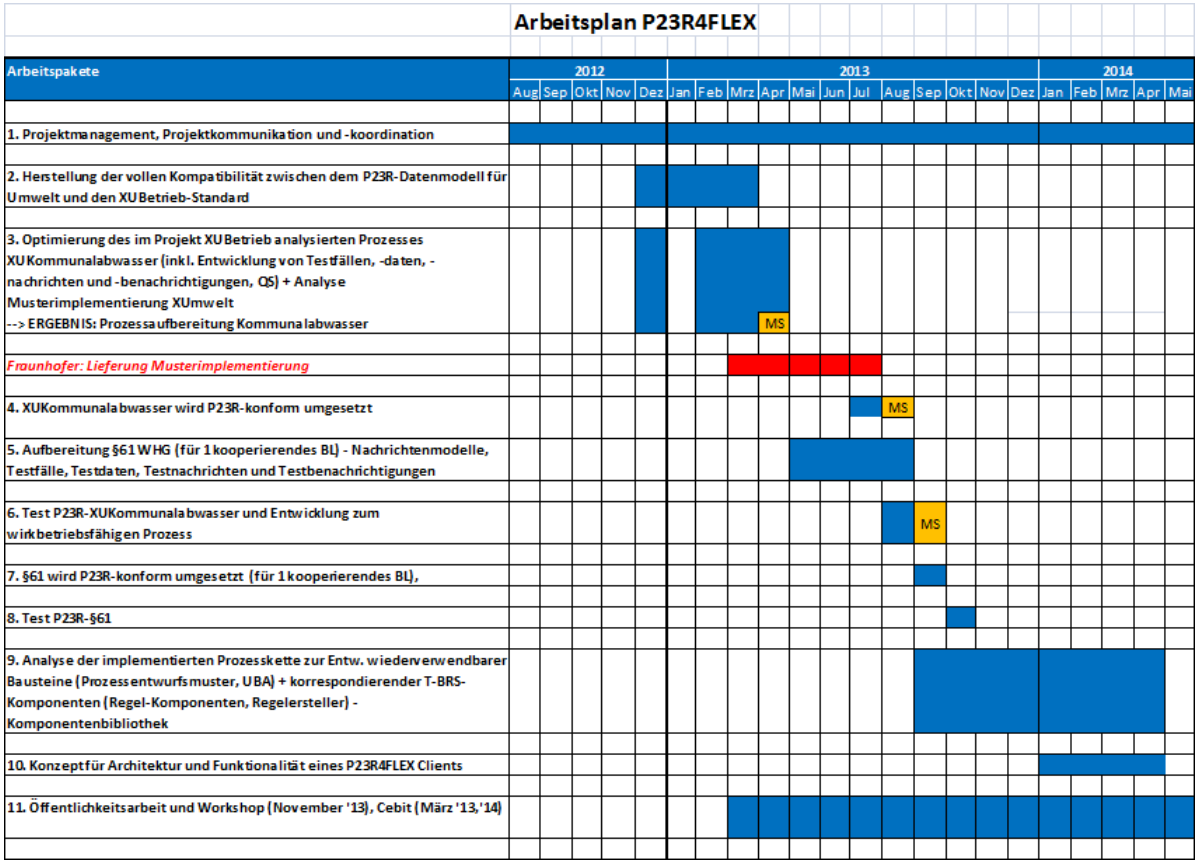


Abbildung 2.1.: Projektplan

Auf einer Projektsitzung am 19.9.2013 wurde erstmals in einer Simulationsumgebung die Abwicklung des Berichtsprozesses EU-Kommunalabwasserrichtlinie (Art. 15(4)) mit Hilfe des P23R Prinzips präsentiert. Die Abwicklung beruhte auf der aktualisierten Musterimplementierung des P23R von Fraunhofer Fokus, weiteren Komponenten, die von der Fa. ENDA im Rahmen des laufenden Projektes entwickelt wurden (u.a. Quell- und Kommunikationkonnektoren) sowie der von Atos entwickelten Berichtsregel einschließlich der Prüfungen für die Qualitätssicherungen. Aufgrund aufgetretener Probleme, insbesondere bei der Anpassung der Musterimplementierung an technische Weiterentwicklungen, haben sich gegenüber der ursprünglichen Projektplanung Verzögerungen ergeben, die eine kostenneutrale Verlängerung bis Ende Mai 2014 notwendig machten. Gleichzeitig wurde eine Erweiterung des Vorhabens ausgearbeitet (Teil 2: P23R im Echtbetrieb bei der BASF (seit Januar 2014)).
Die Projektkommunikation erfolgte über verschiedene Kommunikationsmittel. Regelmäßig wurden Projekttreffen mit allen Beteiligten inkl. BMI durchgeführt. Intern wurde mit Hilfe eines Wikis gearbeitet, damit jederzeit der Stand des Projektes sichtbar war und aktuelle wichtige Dokumente griffbereit zur Bearbeitung vorlagen. Für die Außenkommunikation wurde eine Webseite (www.p23r4flex) eingerichtet. Hier wurde über das Projekt informiert, auf wichtige Ereignisse (CeBIT, Workshop) hingewiesen und Ergebnisse veröffentlicht.

2.2. Qualitäts- und Projektmanagement

Interne Kommunikation mit dem Auftraggeber (BMI)

Während des gesamten Projektes fand monatlich ein Jour fixe im BMI statt. Hier wurde der Projektstand vorgestellt, Probleme diskutiert und neue Ideen entwickelt. Im Februar 2013 wurde nach Bearbeitung der ersten drei Arbeitspakete ein Sachstandsbericht an das BMI übermittelt.

Interne Kommunikation mit den Auftragnehmern (ENDA, FhG, Atos)

Zu Anfang des Projekts wurde einvernehmlich vereinbart, neben Telefonaten, E-Mails und Projekttreffen alle wichtigen Projekt- und Arbeitsstände sowie wichtige Dokumente über ein projektinternes Wiki aktuell zu halten und allen Beteiligten zur Verfügung zu stellen. Alle Projektbeteiligten hatten Zugang und ein volles Bearbeitungsrecht im Wiki. Vorteile in der Arbeit mit dem WIKI sind, dass Veränderungen personengebunden und einfach nachvollzogen werden können und gemeinsam an einem Dokument gearbeitet werden kann. Diese Vorteile wurden bei der Erarbeitung des Handbuches zur Regelerstellung durch FhG leider verschenkt, da das Handbuch nicht im WIKI, sondern direkt im Entwicklerportal erstellt wurde. Veränderungen waren somit nicht direkt nachvollziehbar, was für das Nachverfolgen von Arbeitsständen erheblichen Mehraufwand bei UBA und ENDA bedeutete.
Besonders in der Zusammenarbeit mit FhG fanden eine Reihe interner Workshops statt. Dabei konnte nicht immer eine optimale Balance zwischen Personalressourceneinsatz und Problembearbeitung gefunden werden. Hiervon ist jedoch das Bootcamp im Februar 2014 zwischen ENDA und FhG auszunehmen, bei dem es gelang, in kleinster Runde technische Probleme direkt zu bearbeiten.

3. Ergebnisse

Das Ziel des Projektes - Vorbereitung einer breiten Einführung und Anwendung des Prozessdatenbeschleuniger-Prinzips (P23R-Prinzips) in der Kommunikation zwischen Anlagenbetreiber und Behörde - wurde mit dem beschriebenen Projekt erreicht. Dafür wurden durch die Auftragnehmer folgende Grundlagen geschaffen:

- ENDA KG
  - umfangreichen und detaillierte Prozessbeschreibungen (Anhang C.1, D.1)
  - Qualitätsanforderungen, Qualitätsprüfungen (Anhang C.2)
  - Testfälle mit Prozess- und Nachrichtenmodell sowie Testdatensätzen (Anhang C.3)
  - Implementation von P23R-Regeln für die automatisierte Kommunikation von Berichten und die Programmierung der notwendigen Schnittstellen
  - Konzept für die Architektur und Funktionalität eines P23R4FLEX-Clients (Anhang F)
- Atos IT Solution GmbH
  - Entwicklung zweier Regeln zum Prozess Kommunalabwasserrichtlinie (Anhang C); Download ab Juli 2014 unter http://p23r.enda.eu
  - Entwicklung Regel zum Prozess §61 Wasserhaushaltsgesetz (Anhang D); Download ab Juli 2014 unter http://p23r.enda.eu
- Fraunhofer FOKUS
  - Bereitstellung P23R-Infrastruktur (z.B. Musterimplementierung, Entwicklerportal)
- Fraunhofer FOKUS / Atos IT Solution GmbH / ENDA KG
  - Entwicklung wiederverwendbarer Bausteine (Anhang E)
  - Entwurf Handbuch zur Regelerstellung (Anhang E.5)

Die Akzeptanz des P23R wird durch das neue, einheitliche Umweltdatenmodell, zusammengeführt aus dem P23R-Datenmodell für Umwelt und den XUBetrieb Modellkomponenten, gestärkt. Der Einsatz von XÖV-zertifizierten Standards in den Schnittstellen des P23R schafft Planungssicherheit und Vertrauen bei allen Prozessbeteiligten. Bei der Weiterentwicklung des P23R zu einem praxisorientierten Kommunikationssystem kann sowohl auf die identifizierten Prozessmuster als auch auf die QS-Prüfungsmuster zurückgegriffen werden, um wiederverwendbare P23R-Regelkomponenten (Programmmodule) zu erstellen. Die Bereitstellung von P23R-Regelkomponenten mit einer leicht verständlichen Dokumentation und die Bereitstellung einer intuitiv und effizient nutzbaren P23R-Entwicklungsplatform sind wichtige Schritte zur Durchsetzung des P23R-Prinzips.

3.1. Ergebnisdetails im Anhang

Alle Projektergebnisse werden detailliert im Anhang A - H beschrieben. Am Beginn jedes Kapitels wird der verantwortliche Auftragnehmer erwähnt.

3.2. Umweltberichtspflichten als Vorläufer für weitere Berichtsdomänen

Der Bereich Umwelt wurde als Vorläufer für ein Pilotprojekt im Bereich der Berichtspflichten ausgewählt, da das UBA ein starker Partner in der zukunftsorientierten Prozessgestaltung ist, die technischen und personellen Erfahrungen aus Berichtsprozessen mitbringt und der Bereich Umwelt aufgrund seiner Prozesse gut geeignet ist: die Berichtspflichten sind zum Teil sehr umfangreich mit langen Prozessketten, umfangreichen Datenmengen und vielen Prozessbeteiligten. Sobald die Funktionalität des P23R bei solchen Beispielen nachweisbar ist, ist die Übertragung auf andere Berichtspflichten (z.B. im Bereich Human Resources) leichter realisierbar.
In einem internen Workshop (FhG, ENDA, Atos) wurden erste Ideen für wiederverwendbare Bausteine für zukünftige Regeln zusammengetragen und in einer Mindmap skizziert. Basierend auf den Ergebnissen eines Workshops wurde ebenfalls ein Handbuch zur Regelerstellung zusammengestellt (siehe Anhang E.2).

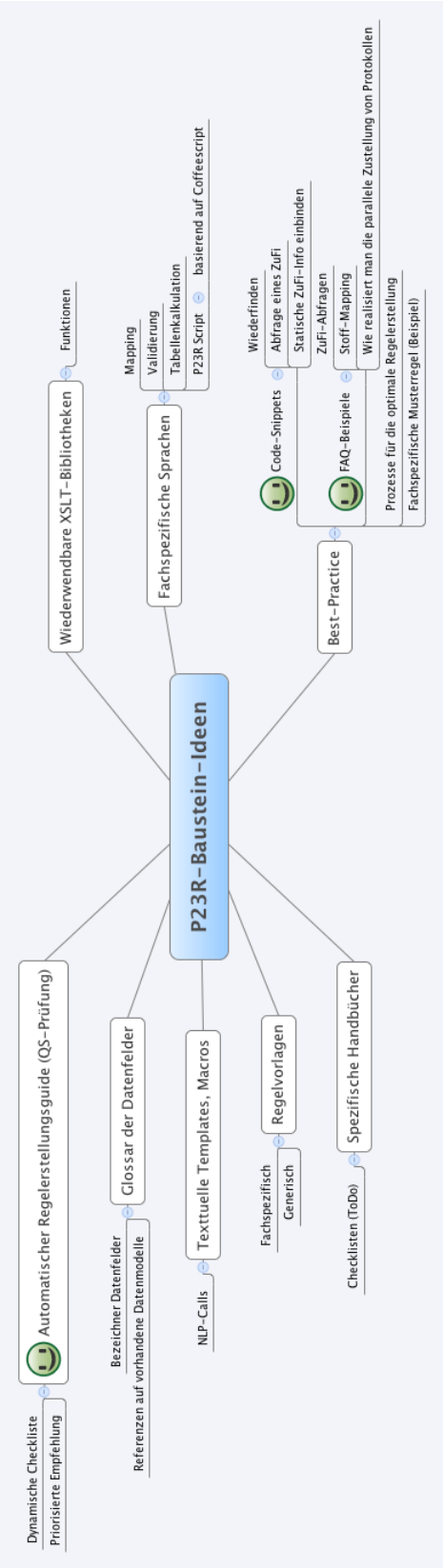


Abbildung 3.1.: Mindmap: Ideen zu wiederverwendbaren Bausteinen und Regelkomponenten

Im Zuge der zweiten testweisen Regelerstellung für den Prozess §61 WHG zeigte sich, dass einige Code-Snippets (siehe Anhang E.2.4.1) wiederverwendet werden können. Durch die Erarbeitung einer Bibliothek für Regelbausteine (siehe Anhang E) wird die Arbeit erleichtert. Eine Übertragung der Regel §61 WHG auf andere Bundesländer wäre ebenfalls sehr einfach zu realisieren, da in einer Reihe von Bundesländern die Struktur der Berichtserstattung an die Betriebstagebücher für Kläranlagen angelehnt ist. Im Projekt wurde deutlich, dass bereits bei der zweiten Regelerstellung der Arbeitsumfang und damit die Bearbeitungszeit deutlich reduziert werden konnten.
Die zukünftige Entwicklung einer größeren Zahl produktionstauglicher P23R-Regeln wird Aufwände nach sich ziehen, die sich durch den Einsatz dokumentierter und wiederverwendbarer P23R-Regelkomponenten senken lassen. Die Untersuchung der Berichtsprozesse für die Erfüllung der Kommunalabwasserrichtlinie 91/271/EWG und der Prozesse im Zusammenhang mit der Selbstüberwachung von Abwasseranlagen gemäß §61 Wasserhaushaltsgesetz (WHG) haben die in Kapitel B.2 Prozessmuster A, B und C als wiederkehrende und über die triviale Übermittlung Sender-Empfänger hinausgehende Teile der Berichtsprozesse identifiziert.
Für die zukünftige Entwicklung dokumentierter und wiederverwendbarer P23R-Regelkomponenten wird empfohlen, zunächst die regelmäßig benötigten, trivialen Operationen

- entgegennehmen einer Nachricht
- auslösen einer Nachricht (Trigger)
- senden einer Nachricht
- umwandeln einer Nachricht

und die Prozessmuster (siehe Anhang D.2.4.) zu unterstützen. Prozessmuster sollten dafür die aufgeführten und strukturierten Qualitätssicherungsmaßnahmen durch P23R-QS-Regelkomponenten unterstützen.

3.3. Zusammenfassung der Ergebnisse aus Sicht des Auftragnehmers ENDA

Das Ziel, Berichtsprozesse aus dem Umweltbereich unter Anwendung des Prozessdatenbeschleuniger-Prinzips (P23R-Prinzips) umzusetzen, ist erreicht worden. Darüber hinaus wurden weitere wichtige Ergebnisse erzielt:

- Die IT-Infrastruktur des Umweltbundesamts ist für die Übermittlung von Berichtsdaten gemäß Kommunalabwasserrichtlinie mit dem P23R vorbereitet. Die Dienste sind produktiv getestet und können jederzeit genutzt werden.
- Es wurde ein neues Pivotdatenmodell für den Abwasserbereich entwickelt, dass XUBetrieb-konform ist und den Berichtsumfang des §61 Wasserhaushaltsgesetz (WHG) abdeckt.
- Für die Qualitätssicherung von §61 WHG-Daten konnten erstmalig zahlreiche Prüfungsvorschriften entwickelt und dokumentiert werden.

- Es wurde ein realistisches Konzept für ein P23R4FLEX-Client-Toolkit erarbeitet, das die Erstellungkosten der noch fehlenden Client-Komponente zur Sichtung und Freigabe der Berichtsdaten für den Breiteneinsatz des P23R deutlich senkt.
- Es liegt eine Analyse vor, die effizientere Validierungsverfahren für XML-Dokumente identifiziert hat.
- Es wurde eine Demonstrationsumgebung mit zwei P23R-Prozessoren und zwei Fachsystemen erarbeitet, die für zukünftige P23R-Vorhaben als Ausgangsbasis verwendet werden kann.
- Die P23R-Musterimplementierung wurde in erheblichem Maße für die Nutzung im realen Einsatz erweitert.

Die für den praktischen Gebrauch des P23R-Prinzips notwendigen Informationen liegen größtenteils in einer Dokumentation von beachtlichem Umfang. Der Nutzen, der dadurch entsteht, dass die Basis der P23R-kundigen IT-Entscheider und Entwickler verbreitert wird, ist erheblich.
Derzeit wird das im Projekt erarbeitete Wissen in praktisch verwertbarer und kompakter Form in eine offen zugängliche und moderne elektronische Kommunikationsplattform gespeist.
Gleichzeitig wird in einer 2. Projektphase die Berichterstattung von vier weiteren Umweltberichtspflichten für die Metropolregion Rhein-Neckar in Zusammenarbeit mit einem großen Industrieunternehmen unter Verwendung des P23R-Prinzips und des vorliegenden P23R-Prozessors automatisiert. Mit diesen Maßnahmen und den bereits erreichten Fortschritten wird eine hohe Akzeptanz für den P23R geschaffen, die eine Senkung der Bürokratiekosten durch eine breite Umstellung auf elektronische Berichtsprozesse möglich macht.

4. Empfehlung und zukünftige Entwicklung

Der dem P23R zugrunde liegende Ansatz stellte sich als richtige, realisierbare und praxistaugliche Idee dar. Einige der im ursprünglichen P23R-Prinzip ausspezifizierten, technischen Randbedingungen für eine praxisorientierte Anwendung haben sich jedoch bei komplexeren Berichtspflichten nicht als die optimale Lösung darstellt. Diesbezüglich sollte die P23R-Spezifikation einem Review, unter Berücksichtigung der Praxiserfahrungen, unterzogen und ggf. angepasst werden.

Was ist wichtig für eine erfolgreiche Einführung von P23R

Neben den folgend genannten Hinweisen sind in den Anhängen zu diesem Bericht detaillierte Informationen zur Regelerstellung, Schemasprachen und Client gegeben (Anhänge E und F (aus Abschlussbericht des Auftragnehmers ENDA KG): Konzept für einen Client; Anhang H3.1 (Abschlussbericht des Auftragnehmers Fraunhofer Fokus /Atos): Informationen zur Erstellung von P23R-Regeln und der Nutzung des Entwicklerportal und der Testumgebung).

- Randbedingungen, Technologie und Infrastruktur
  - Die Musterimplementierung der FhG ist momentan die einzige, von der Zielsetzung offene Implementierung wichtiger Funktionsblöcke des P23R-Konzeptes. Allerdings ist noch nicht klar, unter welcher freien Lizenz sie letztlich veröffentlicht wird. Die Klärung der Lizensierungsfrage ist aber eine Grundvoraussetzung, um eine offene Weiterentwicklung dieser Plattform zu ermöglichen.
  - Wesentliche Vorteile des P23R-Prinzips für die berichtspflichtigen Betriebe sind die einer Regel zugrundeliegende, behördlich autorisierte Spezifikation und die Festlegung von Berichtsinhalten, Qualitätsanforderungen und Rahmenbedingungen einschließlich einer zeitnahen Anpassung von möglichen Änderungen. Dies ist in den Dauerbetriebsszenarien entsprechend vorzusehen und zu berichten.
  - Für die Definition vereinheitlichter Berichtsformate hat sich die Nutzung von (standardisierten) Komponenten für die Schnittstellendefinition (in diesem Falle XUBetrieb (XöV-Standard)) als wichtig und zielführend gezeigt.
  - Eine kritische Mindestmenge von Regeln für verschiedene Berichtspflichten sind zu schaffen (möglichst aus einem Bereich (bspw. Umwelt)), um so interessierten Betreibern und Behörden die Chance zu geben, die anfängliche Investition in einem betriebswirtschaftlich realistischen Zeitrahmen wieder einzuspielen.

  - Eine breite Anwendung des P23R-Prinzips in der Berichterstattung impliziert dessen Anwendung auch bei KMUs, die nicht zusätzlich in Entwicklungskosten investieren können oder wollen. Die Erstellung eines preiswerten, nutzerfreundlichen und komplett einsetzbaren Clients (bzw. eines Client-Toolkits zur Erstellung eines individuellen Clients) zur Bedienung eines P23R ist deshalb von großer Wichtigkeit.
  - Eine Formulierung von QS-Anforderungen ist in dem bisherigen, technischen Regelkonzept (XML-Schema zur Validierung und XSLT für die Transformation und semantische Prüfungen) möglich, aber relativ aufwendig. Im Rahmen dieses Projektes wurde überblicksweise untersucht, ob andere Validierungs-Technologien aus dem XML-Umfeld nach ISO/IEC 19757 (XML-Schema Version 1.1, RELAX NG, SCHEMATRON und DSD) diese Aufgaben vereinfachen können. Das Ergebnis dieser Analyse zeigt, dass eine komplementäre Lösung auf Basis von RELAX NG und SCHEMATRON eine erfolgversprechende Kombination ist. Es wird daher empfohlen, diese Kombination in einem weiteren Projekt vergleichsweise bei einer bekannten Berichtspflicht einzusetzen, um dieses Verbesserungspotential genauer auszuloten.

- Vorteile für Betreiber und Behörden durch den Einsatz von P23R liegen u.a. in einer vereinfachten Datenhaltung und -pflege sowie in qualitativ hochwertigeren Daten, die (im Umweltbereich) in eine bessere (Umwelt)Politik münden können bei gleichzeitig geringeren Kosten für die Datenbereitstellung. Nicht immer sind diese Vorteile allen Beteiligten gleichermaßen ersichtlich und wichtig, sicherlich auch, weil bei allen Beteiligten Anfangsinvestitionen getätigt werden müssen. Bemühungen sind daher nötig, um zu verstärken:
  - das Interesse der Behörden an besseren Daten (durch eine frühzeitig in den Berichtsprozess integrierte Qualitätssicherung),
  - das Interesse bei Betreibern an einer vereinfachten Datenhaltung (einmal erhoben, mehrfach verwendet) und der damit verbundenen, einfacheren und kostengünstigeren Datenpflege:
    * Voraussetzung: Stringenz in der Datenhaltung; Interesse an frühzeitiger Qualitätssicherung,
    * Ergebnisse: weniger Arbeit in der Folge, bessere Eigenüberwachung, Entlastung bei Interpretation gesetzlicher Vorschriften.

- Einstiegshürden für Anwender und Infrastrukturbereitsteller lassen sich merklich reduzieren, wenn:
  - weit verbreitete Werkzeuge genutzt werden (keine Spezialanwendungen, keine neue Sprache etc.),
  - Regeln/ Spezifikationen kostengünstig zur Verfügung stehen (bspw. durch ein Abo-Modell),
  - offene Lizenzen für Schnittstellen und Regeln existieren,
  - keine Kosten für die P23R-Instanz entstehen und eine kostengünstige Konnektorenprogrammierung bei Betreibern und unteren Behörden möglich ist,

  - domänenspezifische Leitstellen geschaffen werden (je Berichtsbereich; z.B Berichtsbereich Umwelt beim UBA) und die Regel- / Spezifikationsverantwortung bei den in einem Berichtsprozess involvierten Behörden liegt,
  - die gesetzliche Verpflichtung, Daten elektronisch und in einem vorgeschriebenen Format zu liefern, generell (z.B. im E-Governmentgesetz) und nicht nur in einzelnen Fachgesetzen (z.B. PRTR-Gesetz) formuliert würde,
  - gut verständliche und kurze Arbeitsdokumentationen frei verfügbar sind.
- Eine kontinuierliche und zielgerichtete Öffentlichkeitsarbeit ist Grundvoraussetzung für eine breite Einführung des P23R-Prinzips in die Praxis. Dazu sind notwendig und zu befördern:
  - Commitment ganz oben in der Politik und Wirtschaft,
  - enge Kooperation mit Industrie und Landesbehörden,
  - weitere erfolgreiche wirkbetriebsnahe Pilotvorhaben - was kann der P23R und wo entstehen dem Unternehmen/Behörden Kosten,
  - regional abgegrenzte Breiteneinführung (Konzept der Erprobungsräume der MRN),
  - viel Überzeugungsarbeit nötig (Interessen IT-Abteilung und Facheinheit nicht unbedingt identisch),
  - Wirtschaftlichkeitsbetrachtungen.

Was nimmt UBA aus diesem Projekt organisatorisch mit?

Die folgend aufgezählten Erfahrungen sind nützliche Hinweise für andere Projekte oder auch für den Teil 2 des P23R4FLEX-Projketes (Wirkbetriebsfähigkeit mit einem Partner aus der Praxis).

- Kommunikation und Hilfsmittel - im Vorfeld verbindliche Einigung auf Werkzeuge (z.B. Veränderungen an Texten müssen Nachvollziehbar sein),
- Beschränkung großer Treffen (zeit- und kostenintensiv), lieber kleine praktische bootcamps,
- Überarbeitung des P23R-Konzepts und der Spezifikation hinsichtlich der in den Pilotprojekten gemachten Erfahrungen sinnvoll (kann neue Software oder neue Verfahren eingesetzt werden?), dies kann auch ein Aufweichen/ Erweitern der Spezifikation bedeuten,
- kontinuierliche Überarbeitung des nutzerfreundlichen Handbuchs zur Regelerstellung.

Während des Projektes wurden eine Vielzahl an Erfahrungen aus Gesprächen, Präsentationen, Arbeitsgruppen, Workshops gewonnen. Die Funktionsfähigkeit der durch Fraunhofer Fokus bei Projektstart bereitgestellten Infrastruktur wurde durch interne Gespräche und Workshops im Projektverlauf verbessert. Interne Arbeitstreffen in großer Runde haben gezeigt, dass viel Zeit für Diskussionen verloren ging. Wesentlich effektiver waren Bootcamps, um Probleme und Fragen direkt vor Ort zu besprechen und zu beheben. Ein Workshop im Herbst 2013 mit Interessierten aus der

Industrie und den Behörden zeigt, dass ein Interesse am P23R gegeben ist. Jedoch ist die Bereitschaft, selbst aktiv zu werden und sich in den Prozess – auch finanziell – einzubringen, bisher verhalten. Hier wird eine gute Öffentlichkeitsarbeit in Zukunft gefragt sein und auch der Ansatz der Erprobungsräume der MRN erscheint sehr sinnvoll.
Auf der CeBIT 2014 konnte erstmalig eine funktionierende Berichtsdatenübertragung mittels P23R gezeigt werden, welches bei den Besuchern auf reges Interesse stieß. Auch hier wurde deutlich, dass zukünftig wirkbetriebsnahe Pilotvorhaben und der reale Einsatz in Unternehmen wichtig sind, um die Funktionsfähigkeit des P23R zu demonstrieren und Unternehmen und Behörden von den Vorteilen zu überzeugen.

Zukünftige Entwicklung

Phase II des Projektes P23R4FLEX

Mit der Demonstration der Praxistauglichkeit des P23R Prinzips in einer weiteren Phase des Projektes P23R4FLEX (bis Herbst 2014, Projektpartner BASF) werden wichtige Erfahrungen für die breite Praxiseinführung des P23R gewonnen. Weitere wichtige Ziele des Vorhabens sind die Identifizierung von Werkzeugen, um künftige Regelentwicklungen zu vereinfachen, die Demonstration der Vorteile, die sich durch eine frühzeitige Integration der Qualitätssicherungsregeln in den Berichtsprozess ergeben sowie der Nachweis der Praxistauglichkeit des P23R-Prinzips auch bei sehr umfangreichen Datenmengen. Durch diesen Projektteil, in dem erstmalig eine produktive Nutzung des P23R-Prinzips durch eine freie P23R-Implementierung erfolgt, werden wichtige Ergebnisse für eine Überführung des P23R-Prinzips in die breite Praxis erzielt. Darauf aufbauen sollte ab 2015 eine Erweiterung der P23R-Nutzung bei diesen konkreten Berichtspflichten um weitere Betriebe bzw. weitere Bundesländer als auch die Umstellung weiterer Berichtspflichten auf das P23R-Prinzip erfolgen. Folgende Ergebnisse sollten als Bausteine unter einer geeigneten Open Source Lizenz veröffentlicht werden:

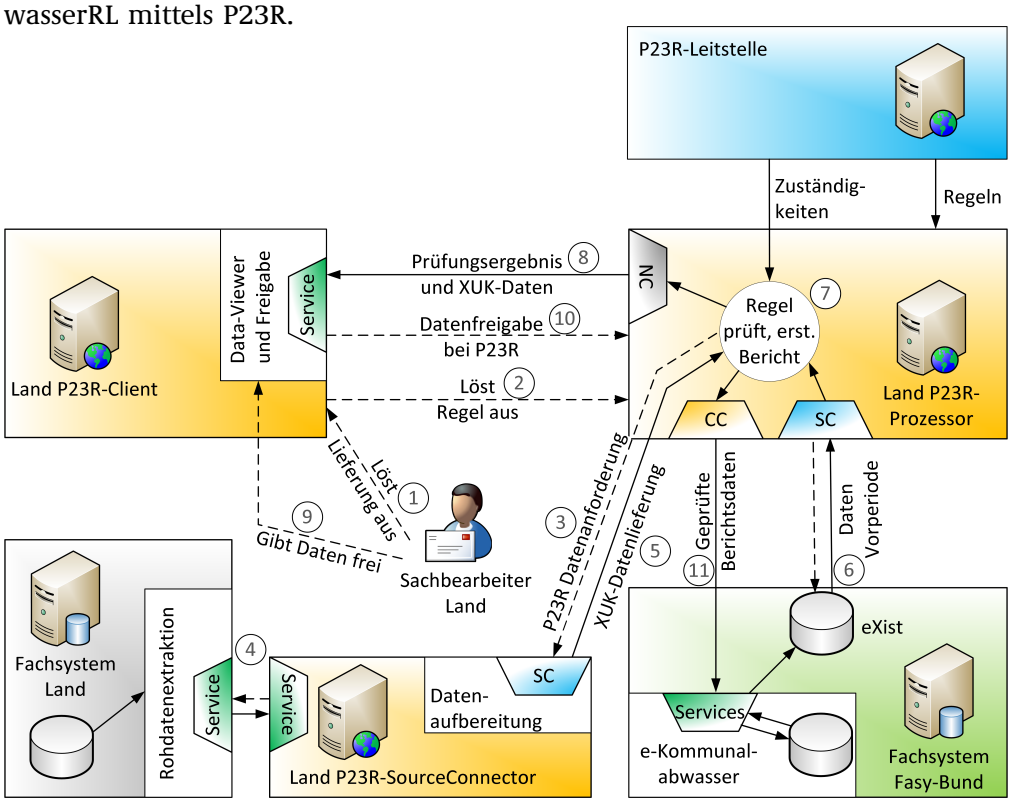
- anwenderorientierte P23R-Clients, die die Regel auslösen, die Berichtsdaten präsentieren und beim P23R freigeben werden können,
- datenlieferantenspezifische P23R-SourceConnectoren, die Daten aus den Fachsystemen automatisiert bereit stellen,
- P23R-Schnittstelle der datenempfangenden Systeme (BuBE-online und E-Kommunalabwasser) (diese sind künftig auch für Lieferungen anderer Betreiber bzw. Länder direkt nutzbar).

Durch diese Arbeiten und deren Veröffentlichung ist der Bestand an Werkzeugen gesichert, der zusammen mit der Musterimplementierung die Realisierung weiterer Berichtspflichten via P23R wesentlich erleichtern wird.

Idee zur Umsetzung der KommAbwRL

Im Bereich der Kommunalabwasserrichtlinie sollte der P23R ebenfalls praxisnah getestet werden. Für die reale Abwicklung von Datenlieferungen sollte die aktuell lauffähige Version des P23R soweit ertüchtigt und die entsprechenden Regeln formuliert, dass unter Verwendung der vorliegenden Regel für die Berichterstattung zur

EU-Kommunalabwasserrichtlinie der anstehende Bericht von einem Bundesland an das Umweltbundesamt über den P23R geliefert wird. Zur Erfüllung dieser Aufgabe sollte bei der zuständigen Behörde des Bundeslandes ein P23R System einschließlich neuer Quelldatenkonnektoren und fachnutzertauglichen Clients installiert werden. Die folgende Grafik verdeutlicht die Projektidee zur Umsetzung der KommunalabwasserRL mittels P23R.



Industrieemissionsrichtlinie (IE-Richtlinie)

Mittelfristig steht die betriebliche Umweltdatenberichterstattung durch die Umsetzung der 2010 von der EU beschlossenen Industrieemissionsrichtlinie (IE-Richtlinie) vor einer deutlichen Veränderung und Erweiterung. Art. 72 dieser Richtlinie legt eine umfassende elektronische Berichtspflicht "über Emissionen und sonstige Arten von Umweltverschmutzung, über Emissionsgrenzwerte, über die Anwendung der besten verfügbaren Techniken" fest. Somit sind für die betroffenen etwa 20.000 Unternehmen in Deutschland Kosten und Aufwände vorprogrammiert, ebenso ein enormer Mehraufwand bei Behörden. Im Entwurf zum deutschen Artikelgesetz zur IE-Richtlinie wird jedoch auf eine Übermittlung der Daten analog den rechtlichen Rahmenbedingungen und der Praxis beim PRTR gesetzt. Ziel sind effektive, medienbruchfreie elektronische Berichtsketten zur Vermeidung von Bürokratiekosten. Somit könnte an dieser Stelle bei der Definition von Prozessen, Datenmodellen und Schnittstellen für die neue IE-Richtlinie auf Erfahrungen und Ergebnisse aus innovativen e-Government-Projekten zurückgegriffen werden und Fehler vermieden werden.

Anhang

A. Herstellung der Kompatibilität zw. dem P23R-Datenmodell für Umwelt und dem XUBetrieb-Standard

Das folgende Kapitel ist dem Abschlussbericht der ENDA KG entnommen.

A.1. Motivation

Für eine erfolgreiche Anwendung des P23R-Prinzips im Umweltdatenbereich musste ein semantisch eindeutiges Datenmodell vorliegen. Mit der vom Projekt XUBetrieb im Auftrag des Umweltbundesamtes entwickelten Bibliothek fachlicher Modellkomponenten lagen frei verfügbare, strukturierte und dokumentierte Datenstrukturen zur Verwendung in Fachanwendungen vor. Die Veröffentlichung der Modellkomponenten als zertifizierter XÖV-Standard gewährleistete gleichzeitig für zukünftige Implementierungen im Umweltdatenbereich Investitionsschutz und Zukunftssicherheit. Aufgrund der organisatorischen Rahmenbedingungen wurden die XUBetrieb Modellkomponenten zeitgleich mit der fachlichen Spezifikation für die Fachdomäne Umwelt/Emissionen als Teil des Pivot-Datenmodells im Kontext der P23R Rahmenarchitektur im Rahmen des Projekts P23R erarbeitet. Bisherige Realisierungen des P23R-Prinzips konnten daher nicht auf die XUBetrieb Komponentenbibliothek zurückgreifen.

A.2. Vorgehensweise

Idealerweise werden für zukünftige Vorhaben beide Konzepte genutzt und zu einer Einheit verbunden. Dazu wurde das P23R-Datenmodell für Umwelt auf der Grundlage von XUBetrieb remodelliert. Für eine bestmögliche Wiederverwendung wurde das bekannte, in der anerkannten Notation UML2 bereitgestellte Modell des XUBetrieb-Standards genutzt. Die plattform- und technologieunabhängige Modellierung in UML2 stellt ein einheitliches Verständnis bei unterschiedlichen Zielgruppen sicher und ist gleichzeitig die Ausgangsbasis für die automatisierte Transformation des remodellierten P23R-Datenmodells in das im vorliegenden Dokument spezifizierte XML-Schema. Als Ausgangsbasis wurde das P23R Umweltdatenmodell aus dem Ergebnisdokument „Fachliche Spezifikation Fachdomäne Umwelt“ (Stand 27.11.2012) und dem dort referenzierten XMLSchema „XUEmission2-0“ verwendet. Ausschlaggebend war dabei das Ergebnisdokument. Zielmodell für die angestrebte Abbildung war die UML-Modellierung der vereinigten betrieblichen Umweltdatenmodelle XUBetrieb Version 1.0.0.

A.3. Mapping

Das Mapping des P23R Umweltdatenmodells konnte sicher gestellt werden. Angelehnt an die Vorgaben von XÖV konnten 2/3 aller Attribute (entspricht xs:element im korrespondierenden Schema) der im P23R Umweltdatenmodell definierten Klassen (entspricht xs:complexType im korrespondierenden Schema) direkt durch XUBetrieb-Modellkomponenten dargestellt werden. Die nicht direkt darstellbaren Attribute wurden, wie in XÖV vorgeschlagen, im Namensraum des konkreten Standards unter Integration abgeleiteter Modellkomponenten als UML Modell umgesetzt.
Beim Zuschnitt der Klassen, welcher durch XUBetrieb vorgegeben ist, konnte keine vollständige Deckung erzielt werden. Datentypen konnten entweder auf den selben Datentyp oder auf einen allgemeineren abgebildet werden. Weitere Ergebnisdokumente sind das unter Nutzung des XGenerators aus dem UML Modell erzeugte XML Schema und die korrespondierende Spezifikation.

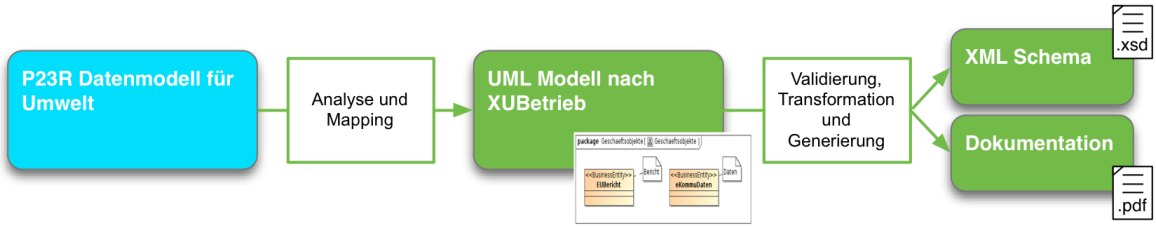


Abbildung A.1.: Vorgehen und Artefakte beim Umweltmodell

B. Prozessoptimierung

Das folgende Kapitel ist dem Abschlussbericht der ENDA KG entnommen.
Die Prozessoptimierung diente der Vorbereitung der Umsetzung des elektronischen Nachrichtenaustauschs nach dem P23R-Prinzip. Ziel war dabei die Beschreibung der Kommunikationsprozesses für die EU-Richtlinie 91/271/EWG Kommunalabwasser und der Selbstüberwachung von Abwasseranlagen gemäß §61 Wasserhaushaltsgesetz (WHG) in einer etablierten Notation.Weiteres Ziel war es, zu prüfen, ob eine Umgestaltung der Prozesses sinnvoll ist, um die Realisierung einer Kommunikation nach dem P23R-Prinzip zu vereinfachen. Mit Blick auf die Automatisierung zahlreicher weiterer Kommunikationsprozesse der betrieblichen Umweltberichterstattung sollten Module für zukünftige P23R Regelimplementationen definiert und in Form von Umweltprozessmustern modelliert werden. Weiteres Ziel war es, die Qualitätssicherungsanforderungen (QS-Anforderungen) zu untersuchen und zu kategorisieren, um Module für zukünftige P23R Regelimplementationen zu definieren. Ferner sollten QS-Anforderungen so detailliert beschrieben werden, dass eine entsprechende Regelimplementationen durchgeführt werden konnte.
Die bisherigen Untersuchungen von Prozessen im Bereich der Umweltberichterstattung hatten sich auf die Ebenen Bundesland und Bund fokussiert. Bei den im Projekt betrachteten Prozessen war eine Erweiterung bis zur kommunalen Ebene nötig, da hier der Datenursprung lag. Die bisher modellierten Prozesse für die beiden genannten Berichterstattungen wurden daraufhin erweitert und angepasst. Die Bundesländer Sachsen und Bayern sagten Ihre Unterstützung bei der Prozessaufnahme zu. Für die Identifikation von Modulen für zukünftige P23R Regelimplementationen wurde eine Separation von Prozessteilen vorgenommen, von denen angenommen werden konnte, dass sie allgemein gültigen Charakter hatten. Dieses Kriterium galt als erfüllt, wenn ein Prozessteil mehrfach anzutreffen war oder dem Analytiker solche Teile aus anderen Prozessmodellierungen im Bereich betrieblicher Umweltberichterstattung bereits bekannt waren. Mit der Bestandsaufnahme und der Beurteilung des Reifegrads der verfügbaren P23RBestandteile konnten die Projektaufgaben genauer ausgerichtet werden, um den Projektnutzen sowohl für das P23R-Prinzip als auch die Berichtsaufgaben des Umweltbundesamts weiter zu erhöhen. Für die Durchführung von Interviews mit den Fachverantwortlichen eines Bundeslandes wurden die Länder Hessen, Bayern und Sachsen um Kooperation gebeten. Bayern und später Sachsen sagten Bereitschaft zur Unterstützung des Vorhabens zu.
Für die Modellierung wurde UML2 verwendet, da es die dominierende und vielseitigste Sprache für die Softwaresystem-Modellierung darstellt, eine grafische Repräsentation besitzt und elektronisch auswertbar ist. Dabei wurden die Diagrammtypen Aktivitäts- und Anwendungsfall-Diagramme (Use-Case-Diagram) und benutzt.

B.1. Prozessbeteiligte + Prozessorganisation

B.1.1. Bayern

Die Prozessmodellierung zeigt zunächst im Anwendungsfall-Diagramm die Prozessbeteiligten (Abb. B.1) der analysierten und dokumentierten Prozesse.

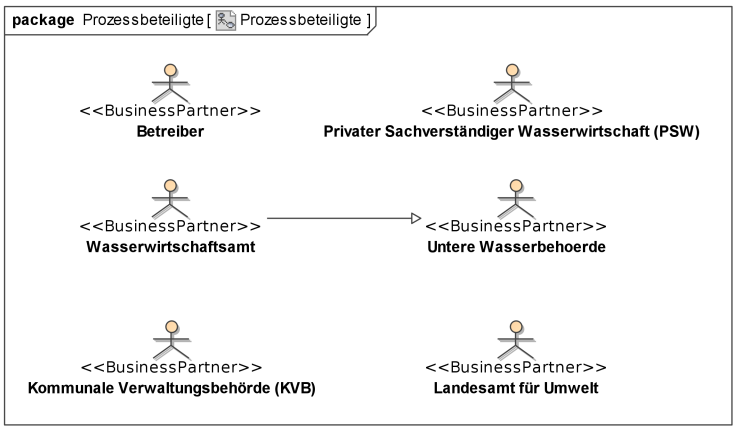


Abbildung B.1.: Use-Case-Diagramm „Prozessbeteiligte §61 WHG, Genehmigungsantrag, KA-Jahresbericht und 91/271/EWG“ Bayern

Der Use-Case „Genehmigungsantrag“ (Abb. B.2) mit den Aktivitäten „Genehmigungsantrag stellen“ (Abb. B.3 und „Genehmigungsantrag bearbeiten“ (Abb. B.4) war nicht direkt mit §61 WHG oder Kommunalabwasserberichterstattung verbunden. Die Aktivitäten stellten jedoch den ersten Schritt bei der Betrachtung eines wasserwirtschaftlichen Objekts dar und wurden deshalb hier betrachtet. In der Aktivität „Genehmigungsantrag bearbeiten“ wurd das wasserwirtschaftliche Objekt erstmalig angelegt. Auf die dabei hinterlegten Informationen wird im Rahmen der 91/271/EWG-Berichterstattung zurückgegriffen, wenn z. B. bauliche Informationen zur Kläranlage benötigt werden.

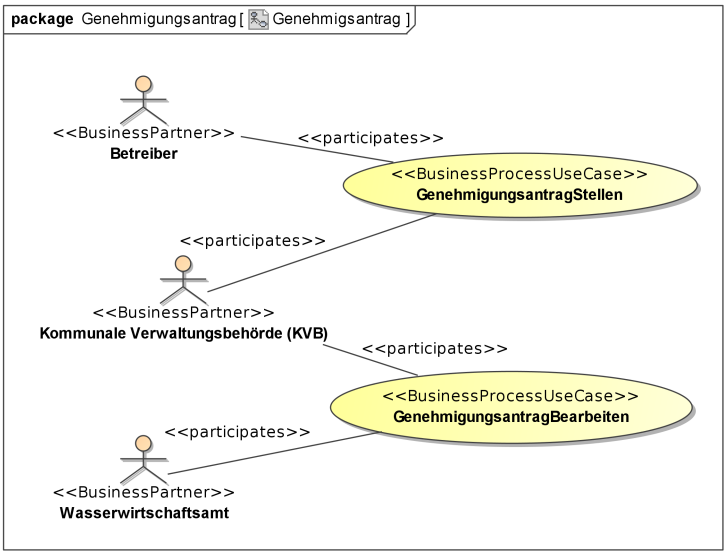


Abbildung B.2.: Use-Case-Diagramm „Genehmigungsantrag" Bayern

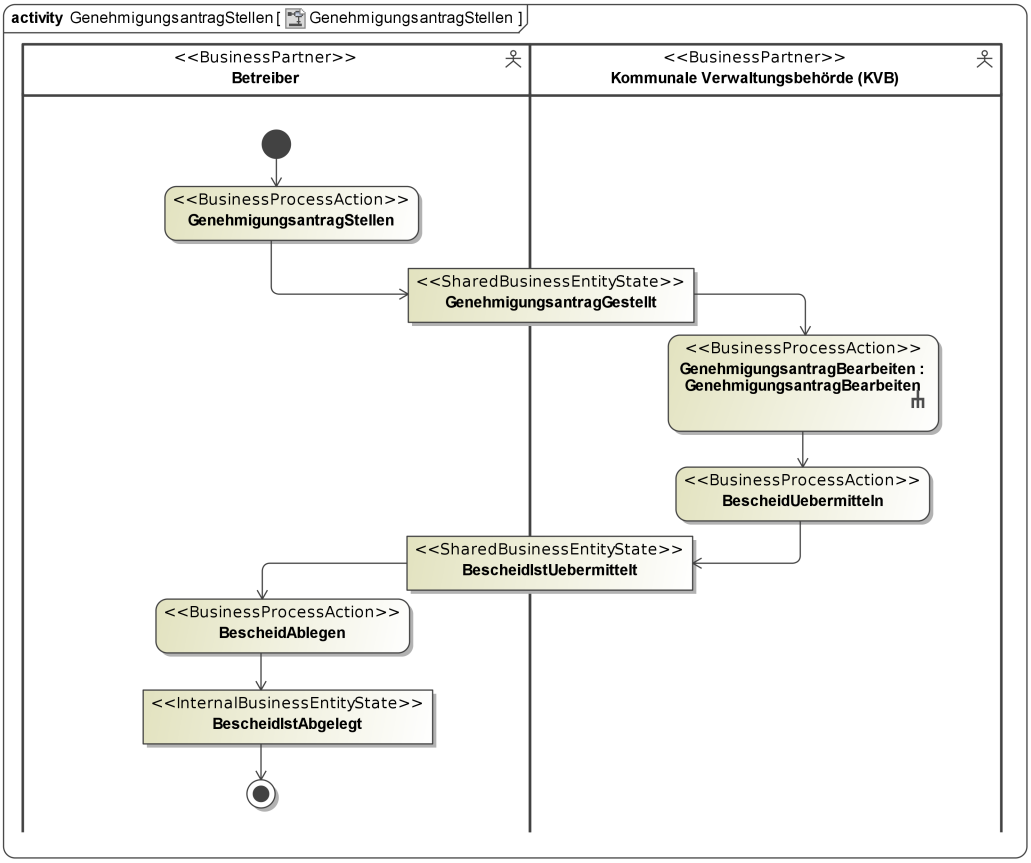


Abbildung B.3.: Aktivitätsdiagramm „Genehmigungsantrag stellen „Bayern

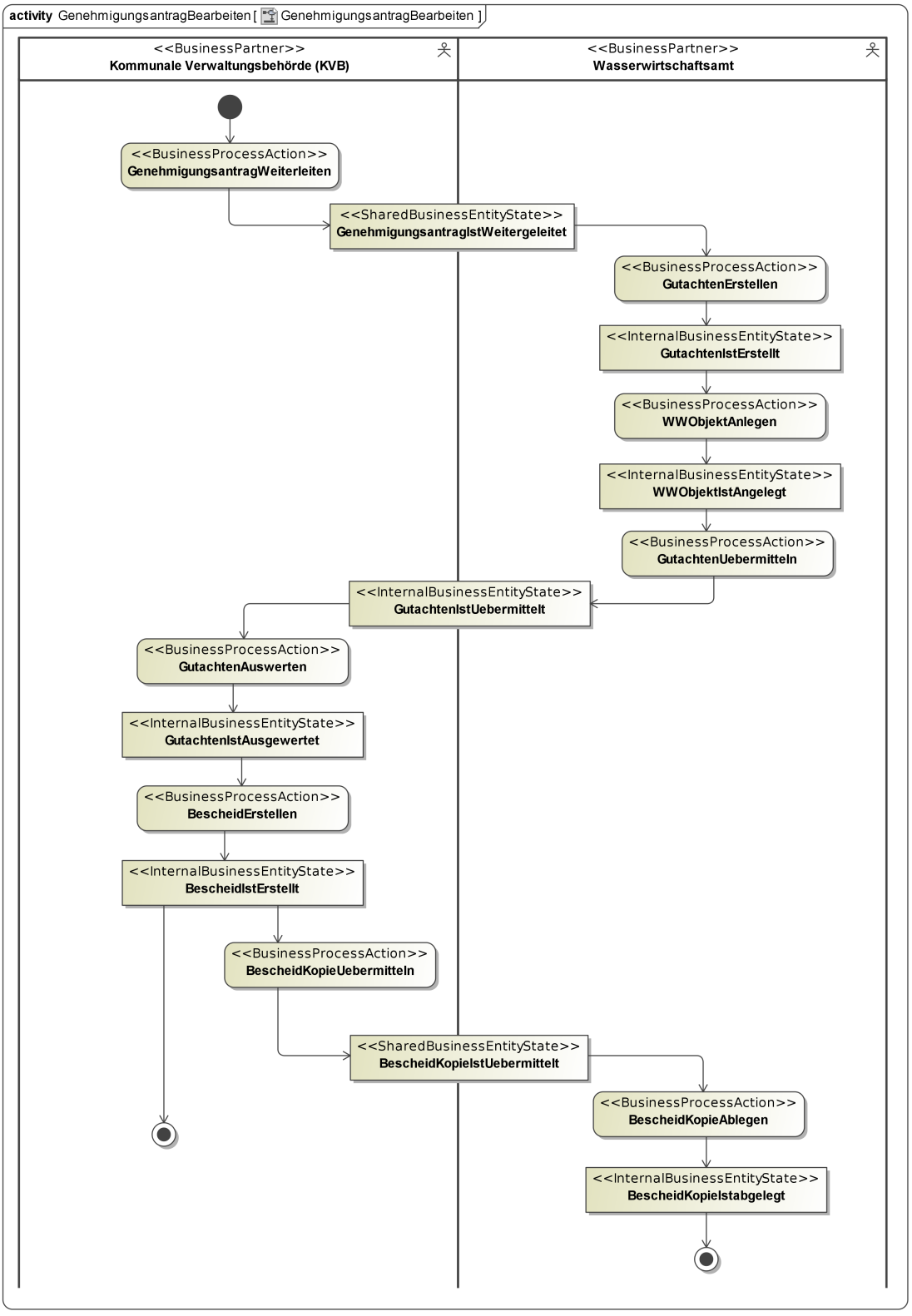


Abbildung B.4.: Aktivitätsdiagramm „Genehmigungsantrag bearbeiten” Bayern

B.1.2. Sachsen

Um eine breitere Basis für die Prozessanalyse im Bereich der Landesbehörden und der unteren Wasserbehörden zu schaffen, wurden auch mit den Fachverantwortlichen der sächsischen Landesdirektion und des sächsischen Landesamts für Umwelt, Naturschutz und Geologie und Interviews geführt. Dabei wurden erwartungsgemäß deutliche Unterschiede in den organisatorischen Abläufen und in der Strukturierung der unterstützenden IT-Systeme Sachsens und Bayerns festgestellt. Daraus resultierten unterschiedliche Anwendungsfälle und Aktivitäten. In der sächsischen Wasserverwaltung halten derzeit drei Datenbanken die wesentlichen Informationen für die Selbstüberwachung der Kläranlagen und die Berichtersta..ungen gemäß EU-Richtlinie 91/271/EWG. Dies waren zum einen

- die Datenbank kommunales Abwasser, die vom Sächsischen Landesamt für Umwelt, Landwirtschaft und Geologie entwickelt und betrieben wird und Informationen über die Siedlungsgebiete (z. B. Einwohner und Einwohnergleichwerte) und Anlagen (z. B. Reinigungsstufen und Arten der weitergehenden Behandlung) hält und aus der die wesentlichen Daten des Lageberichts Abwasser abgeleitet werden,
- das elektronische Wasserbuch FIS-WRV (Fachinformationssystem wasserrechtlicher Vollzug), das die Daten der Einleitung, die Eigenüberwachungsergebnisse und zukünftig auch die behördlich zu überwachenden Werte sowie die behördlichen Überwachungsergebnisse hält und
- die Datenbank Abwasserdirekteinleiterüberwachung (ADÜ), die derzeit noch die behördlichen Überwachungswerte und -ergebnisse hält und zukünftig von einem neu zu entwickelnden FIS-WRV abgelöst wird.

Bereits bei der Betrachtung der Prozessbeteiligten (Abb. B.5) traten Unterschiede zwischen Sachsen und Bayern zu Tage. Der bayerische „private Sachverständige in der Wasserwirtschaft (PSW)“ ist in Sachsen nicht bekannt. Eben sowenig die kommunale Verwaltungsbehörde. Dafür kann die Rolle der zuständigen Behörde in Sachsen sowohl von der unteren Wasserbehörde als auch (bei großen Kläranlagen) von der Landesdirektion wahr genommen werden.

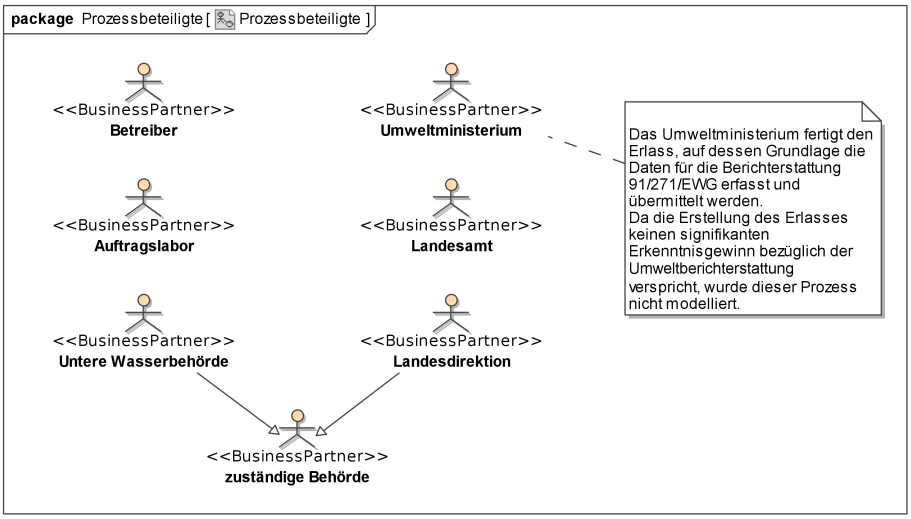


Abbildung B.5.: Use-Case-Diagramm „Prozessbeteiligte §61 WHG und Berichterstattung 91/271/EWG“ Sachsen

Die Akteure und ihre Teilnahme bei den verschiedenen Aktivitäten sind im (Abb. B.6) dargestellt.

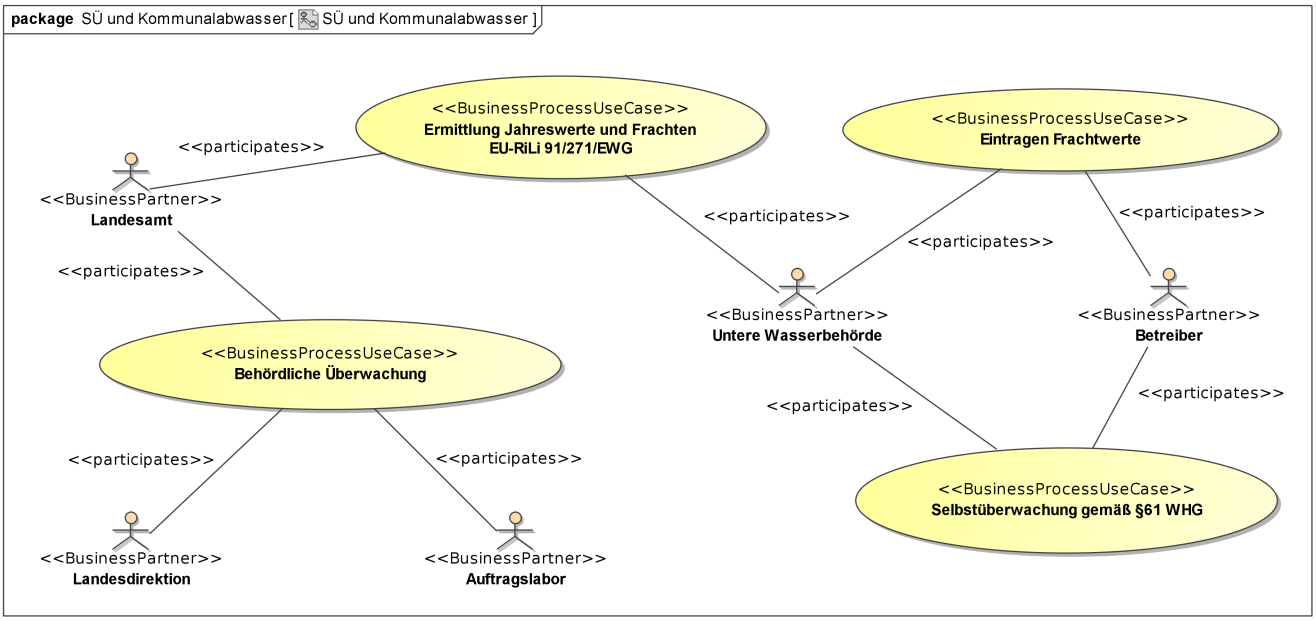


Abbildung B.6.: Use-Case- Diagramm „Selbstüberwachung §61 WHG und Berichterstattung 91/271/EWG“ Sachsen

Die Aktivität „Behördliche Überwachung“ Sachsen (Abb. B.7) war weder Teil der Berichterstattung gemäß EU-Richtlinie 91/271/EWG noch der Eigenkontrollverordnung (EKV). Sie wurde aufgenommen, da sie über die Speisung der Datenbankanwendung ADÜ (Abwasserdirekteinleiterüberwachung) mit Überwachungswerten und -ergebnissen indirekt für die Beantwortung der Frage nach Konformität der CSB-

und BSB5-Emissionen mit den Anforderungen der EU-Richtlinie während der letzten zwölf Monate notwendig ist, ferner Hinweise auf Einzelfallüberschreitungen liefert und indirekt für die Beantwortung der Frage nach Konformität der CSB- und BSB5-Konzentrationswerte mit Anhang 1 der Abwasserverordnung (AbwV) notwendig ist. Die fakultativen Fragen nach den Stickstoff- (N-) und Phosphor- (P-) Konzentrationen der Einleitung können ebenfalls mit Hilfe der bei der behördlichen Überwachung ermittelten Überwachungsergebnisse beantwortet werden.

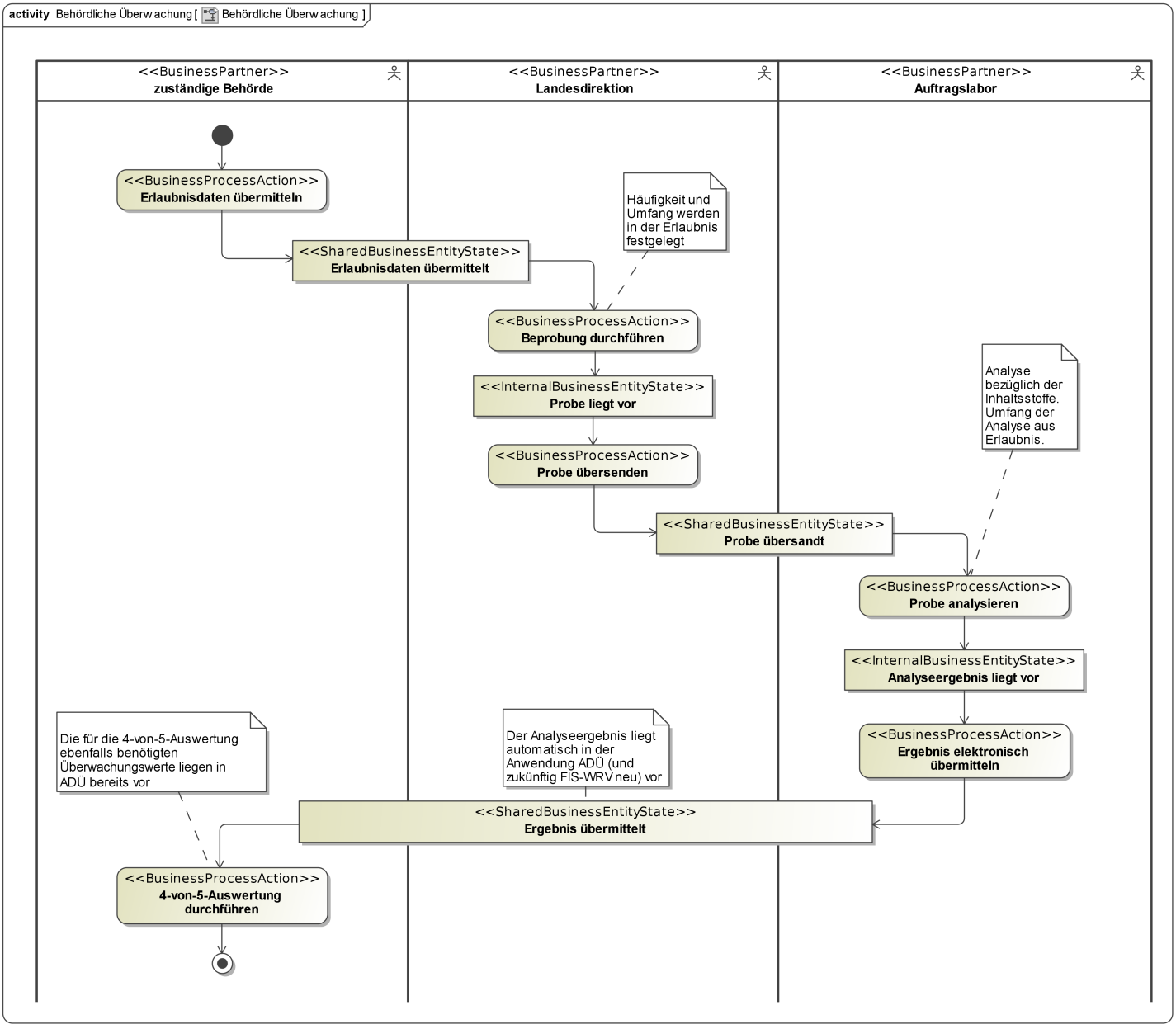


Abbildung B.7.: Aktivitätsdiagramm „Behördliche Überwachung“ Sachsen

B.2. Umweltprozessmuster

Um die Optimierung der eingangs erwähnten zahlreichen weiteren Kommunikationsprozesse effizient und innerhalb eines begrenzten zeitlichen Rahmens vornehmen zu können, war die Untersuchung auf wiederkehrende Prozessmuster als Blaupause für wiederverwendbare Module sinnvoll. Dabei waren jedoch die spezifischen Eigenschaften des P23R-Prinzips zu beachten. Das P23R-Prinzip ist auf die elektronische Übertragung von Nachrichten in vordefinierten XML-Formaten ausgerichtet. Dabei sieht es die Möglichkeit vor, Inhalte zu transformieren. Daneben sieht das P23R-Prinzip eine Triggerfunktion und eine automatisierte Empfängersuche vor.
Die zu untersuchenden Prozesse wurden in Aktivitätsdiagrammen dokumentiert. Zunächst wurden diejenigen Arbeitsschritte aussortiert, die nicht vom P23R-Prinzip abgedeckt wurden. Die verbleibenden Arbeitsschritte waren im fachlichen Kontext für eine P23R-Eignung zu bewerten.

B.2.1. Auswertung der Prozesse für das Land Bayern

Aktivität „Genehmigungsantrag stellen“ (Abb. B.3)

- Die Übermittlung von Genehmigungsanträgen für den Bau von Kläranlagen ist für ein Prozessmuster nicht geeignet, da weder beim Empfänger (kommunale Verwaltungsbehörde) die Voraussetzungen für die Bearbeitung elektronischer Bauzeichnungen vorliegen, noch ein einheitliches Format zur Darstellung und Bearbeitung von Bauzeichnungen rechtlich durchsetzbar ist. Sie werden daher auch weiterhin auf Papier übermittelt werden.

- Die Übermittlung eines Genehmigungsbescheids von der Behörde an das Unternehmen (Betreiber) kann hingegen nach dem P23R-Prinzip funktionieren. Da die Genehmigung von Kläranlagen ein niederfrequenter Vorgang ist und der Genehmigungsbescheid je Kommunalbehörde unterschiedlich aufgebaut sein kann, wird dieser Prozessteil jedoch nicht für eine baldige P23R-Umsetzung empfohlen.

Aktivität „Genehmigungsantrag bearbeiten“ (Abb. B.4)

- Die Weiterleitung von Genehmigungsanträgen ist aus den gleichen Gründen, wie sie bei der Übermittlung von Genehmigungsanträgen für den Bau von Kläranlagen (s.o.) gelten, nach dem P23R-Prinzip nicht möglich.

- Die Erstellung von Gutachten liegt ebenso wie die Anlage wasserwirtschaftlicher Objekte in Fachsystemen außerhalb des Wirkungsbereichs des P23R-Prinzips.

- Die Übermittlung des Gutachtens unterscheidet sich unwesentlich von der Übermittlung eines Genehmigungsantrags und ist nicht für das P23R-Prinzip geeignet.

- Die Übermittlung eines Genehmigungsbescheids von der kommunalen Verwaltungsbehörde an die untere Wasserbehörde (Bayern: Wasserwirtschaftsamt) kann nach dem P23R-Prinzip funktionieren. Da die Genehmigung von Kläranlagen ein niederfrequenter Vorgang ist und der Genehmigungsbescheid je

Kommunalbehörde unterschiedlich aufgebaut sein kann, wird dieser Prozessteil jedoch nicht für eine baldige P23R-Umsetzung empfohlen.

Aktivität „Kläranlagenjahresbericht anfordern“ (Abb. C.3)

- Dies könnte durch Nutzung des P23R-Prinzips unterstützt werden: Ein Fachsystem sieht zum Liefertermin nach, ob alle Betreiber ihren Jahresbericht vorgelegt haben. Ist dies nicht der Fall, wird ein Landes-P23R vom Fachsystem über einen Webservice angesteuert, um die Aufforderungen zur Anforderung des Jahresberichts bei der kommunalen Verwaltungsbehörde auszulösen.
  Sowohl die Aufforderung der unteren Wasserbehörde zur Anforderung des Jahresberichts bei der kommunalen Verwaltungsbehörde als auch die Anforderung des Jahresberichts durch die kommunale Verwaltungsbehörde beim Betreiber können konform zum P23R-Prinzip realisiert werden. Die bei der Übermittlung zu verwendenden Datenstrukturen dürften geringe Komplexität aufweisen und leicht normierbar sein.
  Diese Kombination von Aufforderung und Anforderung zwischen drei Ebenen ist ein über einen einzelnen Prozessschritt hinausgehendes **Prozessmuster (A)**.

- Der Jahresbericht ist nicht formgebunden, könnte aber elektronisch erstellt und übermittelt werden, wenn der Betreiber eine IT-Lösung im Einsatz hat, die über die wesentlichen Informationen verfügt. Diese Kommunikation durch den P23R zu realisieren ist interessant, da beim Betreiber ein Sicherheitsgewinn („Habe ich alles richtig gemacht?“) entsteht. Durch den Einsatz XUBetrieb-konformer Schnittstellen kann eine klare Definition des strukturellen Umfangs erfolgen.

- Wenn die Übermittlung des Jahresberichts vom Betreiber an die kommunale Verwaltungsbehörde nach dem P23R-Prinzip erfolgt, kann und sollte auch die Übermittlung des Jahresberichts von der kommunalen Verwaltungsbehörde an die untere Wasserbehörde nach dem P23R-Prinzip erfolgen. Es weicht einzig der Empfänger ab. Die Regeln könnten unverändert übernommen werden.
  Diese Übermittlung der Jahresberichte zwischen drei Ebenen ist ein weiteres, über einen einzelnen Prozessschritt hinausgehendes **Prozessmuster (A)**.

Aktivität „Kläranlagenjahresbericht erstellen“ (Abb. D.3)

- Aufbereitung und Freigabe des Jahresberichts sowie die Prüfung der Zuständigkeit sind nicht vom P23R-Prinzip abgedeckt.

- Die Übermittlung des Jahresberichts nach dem P23R-Prinzip wurde bereits in der Aktivität „Kläranlagenjahresbericht anfordern“ befürwortet.

- Die Prüfung des Jahresberichts ist bisher nicht spezifiziert, da noch nicht einmal die inhaltliche Struktur spezifiziert ist. Aufgrund der Tatsache, dass ein beträchtlicher Teil des Jahresberichts nummerische Angaben zu Abwasserdurchfluss und Konzentrationen enthält, ist es sicher, dass Prüfungen spezifiziert werden können. Diese werden die Prüfung durch Sachverständige und/oder untere Wasserbehörde nicht vollständig ersetzen. Trotzdem sollten alle Prüfungen, die auf der zu übermittelnden Nachricht ohne Rückgriff auf Altdaten ausgeführt werden können, auch durchgeführt werden. Besser wäre noch, das P23R-Prinzip so zu ertüchtigen, dass zu Prüfzwecken aus dem P23R auf Altdaten zugegriffen werden kann. Damit würden Zeitreihenprüfungen ermöglicht.

- Wenn eine Übertragung des Jahresberichts nach dem P23R-Prinzip zwischen den Kommunikationsteilnehmern etabliert wird, sollte auch die Korrekturbedarfsmeldung nach dem P23R-Prinzip realisiert werden.
- Die Kombination aus Übermittlung des Jahresberichts, Durchführung (eines Teils) der Prüfungen und Übermittlung einer Korrekturbedarfs- oder Erfolgsmeldung ist ein **Prozessmuster (B)**.

Aktivität „Kommunalabwasserberichterstattung 91/271/EWG“ (Abb. C.2)

- Die Anforderung der Kläranlagenbetriebsinformation, die aus dem Jahresbericht und den amtlichen Überwachungsdaten generiert wird, kann nach dem P23RPrinzip erfolgen, wenn die Kommunikationspartner sich auf eine Schnittstellendefinition einigen. Die bei der Übermittlung zu verwendenden Datenstrukturen dürften geringe Komplexität aufweisen und leicht normiert werden können.
- Die Übermittlung der Kläranlagenbetriebsinformation kann nach dem P23R-Prinzip erfolgen, wenn eine XUBetrieb-konforme Definition der Schnittstelle erfolgt. Diese Schni..stelle hat jedoch potenziell sehr viele verschiedene Ausgestaltungen, da es sich um eine bisher inhaltlich nicht spezifizierte Schni..stelle zwischen unterer Behörde (derer gibt es viele) und Landesamt handelt, deren Inhalte durch keine Rechtserkenntnisquelle festgelegt sind. Das Sparpotenzial durch den Einsatz einer IT-Lösung nach dem P23R-Prinzip ist dann außergewöhnlich hoch, wenn keine IT-basierte Lösung für die Kommunikation existiert. Ansonsten ergeben sich in jedem Fall die durch das P23R-Prinzip immer realisierbaren Vorteile. Die Einigung auf möglichst wenige, vorzugsweise nur eine einzige Schnittstellendefinition hängt von der Kooperationsbereitschaft der Kommunikationsteilnehmer ab.
- Die Prüfung der Kläranlagenbetriebsinformation ist bisher nicht spezifiziert. Diese Betriebsinformation wird für die Berichterstattung nach 91/271/EWG beschafft und dürfte daher inhaltlich große Ähnlichkeit zu XUKommunalabwasser aufweisen. Folglich sollte auch ein großer Teil der für die Kommunalabwasserberichterstattung spezifizierten Prüfungen sinnvoll auf diese Kläranlagenbetriebsinformation anwendbar sein. Diese automatisierten Prüfungen werden die Prüfung durch die Fachverantwortlichen des Landesamts jedoch nicht ersetzen.
  Trotzdem gilt: Es sollten alle Prüfungen, die auf der zu übermittelnden Nachricht ohne Rückgriff auf Altdaten ausgeführt werden können, auch durchgeführt werden. Besser wäre noch, das P23R-Prinzip für den Zugriff auf Altdaten zur Durchführung von Zeitreihenprüfungen zu erweitern.
- Wenn eine Übertragung der Kläranlagenbetriebsinformation nach dem P23RPrinzip zwischen den Kommunikationsteilnehmern etabliert wird, sollte auch die Korrekturbedarfsmeldung nach dem P23R-Prinzip realisiert werden.
- Die Kombination aus Übermittlung des Kläranlagenbetriebsinformation, Durchführung (eines Teils) der Prüfungen und Übermittlung einer Korrekturbedarfs- oder Erfolgsmeldung ist dasselbe **Prozessmuster (B)**, das bereits bei der Aktivität „Kläranlagenjahresbericht erstellen“ identifiziert wurde.

B.2.2. Auswertung der Prozesse für das Land Sachsen

Aktivität „Selbstüberwachung gemäß §61 WHG“ (Abb. D.4)

- Die Übermittlung des Jahresberichts nach dem P23R-Prinzip wurde bereits für die gleiche Aktivität im Land Bayern befürwortet. In Kombination mit Aktivität „Eintragen der Frachtwerte“ (Abb. C.5) ergibt sich eine Kombination aus Übermittlung des Jahresberichts, Durchführung von Prüfungen und Übermittlung einer Korrekturbedarfs- oder Erfolgsmeldung und somit wieder das **Prozessmuster (B)**.
  Die inhaltliche Struktur der sächsischen Jahresberichte wird allerdings voraussichtlich von der der bayerischen Jahresberichte abweichen, bis entweder eine bundeseinheitliche Regelung in Kraft tritt oder der Willen zur Nutzung einer IT-Kommunikation mit XUBetrieb-konformen Schnittstellen zu einer faktischen Normierung führt.

Aktivität „Behördliche Überwachung“ Sachsen (Abb. B.7)

- Die Übermittlung von Erlaubnisdaten dürfte bereits IT-gestützt erfolgen. Ist dies nicht der Fall, so kann dies auch gemäß dem P23R-Prinzip erfolgen.

- Eine Übermittlung von Proben auf elektronischem Weg ist auch unter Nutzung des P23R-Prinzips noch nicht möglich.

- Die Analyseergebnisse werden bereits elektronisch übermittelt. Dies dürfte länderübergreifend sowohl bei Landes- als auch bei Auftragslaboren aller Bundesländer inzwischen unter Nutzung von Laborinformations- und -managementsystemen (LIMS) der Fall sein.

Aktivität „Ermittlung Jahreswerte und Frachten EU-Richtlinie 91/271/EWG“ (Abb. C.4)

- Sowohl die Anforderung von Frachtdaten als auch deren Rückübermittlung sind nach dem P23R-Prinzip realisierbar. Unter Einbeziehung von Prüfungen auf den Rückübermittelten Frachtdaten ergäbe sich wieder das bekannte **Prozessmuster (B)**.
  Vor dem Einsatz des P23R-Prinzips ist jedoch zu untersuchen, ob diese Art der Trennung von Frachtdaten einerseits von Abwassermengen und Konzentrationen andererseits ein eher spezifisch sächsisches Vorgehen darstellt – in diesem Fall wäre das Sparpotenzial auf Sachsen begrenzt – oder ob diese Situation auch in anderen Ländern anzutreffen ist.

Aktivität „Eintragen der Frachtwerte“ (Abb. C.5)

- Hier geht es vor allem um die Erfassung von Jahreswerten und Frachten und bei Identifikation unplausibler Werte um die Korrekturanforderung des Jahresberichts. Wie oben erwähnt ergibt sich in Kombination aus Übermittlung des Jahresberichts, Durchführung von Prüfungen und Übermittlung einer Korrekturbedarfs- oder Erfolgsmeldung das **Prozessmuster (B)**.

B.2.3. Auswertung der Prozesse für den Bund

Aktivität „Anforderung EU“ Bund (Abb. C.7)

- Die Übermittlung der Berichtsanforderung der Europäischen Kommission an das deutsche Umweltministerium und die Weiterleitung an das Umweltbundesamt entspricht dem **Prozessmuster (A)** und kann nach dem P23R-Prinzip realisiert werden.
- Die Übermittlung des Berichts in Kombination mit der Durchführung von Prüfungen und der Rückübermittlung einer Korrekturanforderung oder einer Erfolgsmeldung entspricht dem **Prozessmuster (B)** und kann nach dem P23R-Prinzip realisiert werden. Schnittstellenstrukturen und Prüfungen wären dann von der Europäischen Kommission vorzugeben.

Aktivität „Anforderung Bund“ (Abb. C.8)

- Die Anforderung der Berichtsdaten zur Kommunalabwasserrichtlinie 91/271/EWG in Kombination mit der (hier im Subprozess „91_271_EWG“ versteckten) Prüfung der Berichtsdaten und der Rückübermittlung der Berichtsdaten entspricht wieder dem **Prozessmuster (B)**, das nach dem P23R-Prinzip realisiert werden kann.
  Dieser Fall ist aufgrund der umfangreichen bereits voll aus spezifizierten Prüfungen und aufgrund einer bereits vorhandenen, bidirektionalen und XUBetriebkonformen Schnittstelle (XÖV-zertifizierter Standard XUKommunalabwasser) auf Seite des Umweltbundesamts für eine Testimplementierung nach dem P23R-Prinzip gut geeignet.

Aktivität „Korrekturanforderung Bund“ (Abb. C.9)

- Der Teilprozess Korrekturanforderung des Umweltbundesamts, Rückübermittlung der Daten vom Landesumweltamt an das Umweltbundesamt und Prüfung (hier nicht sichtbar) entspricht einem neuen **Prozessmuster (C)**, dessen erste beiden Schritte (siehe Abschnitt B.2.4) nach dem P23R-Prinzip realisiert werden können.
- Auch der Teilprozess innerhalb des Landes aus den Schritten Datenkorrektur (von der unteren Wasserbehörde) anfordern, Korrektur entgegen nehmen und Daten prüfen entspricht dem **Prozessmuster (C)**, dessen erste beide Schritte (Abb. 3.24, Seite 38) nach dem P23R-Prinzip realisiert werden können.

B.2.4. Identifizierte Prozessmuster

Die Prozessbeteiligten und ihre Beteiligung an den Prozessmustern sind im Use-Case-Diagramm „Prozessmuster“ (Abb. B.8) dargestellt.

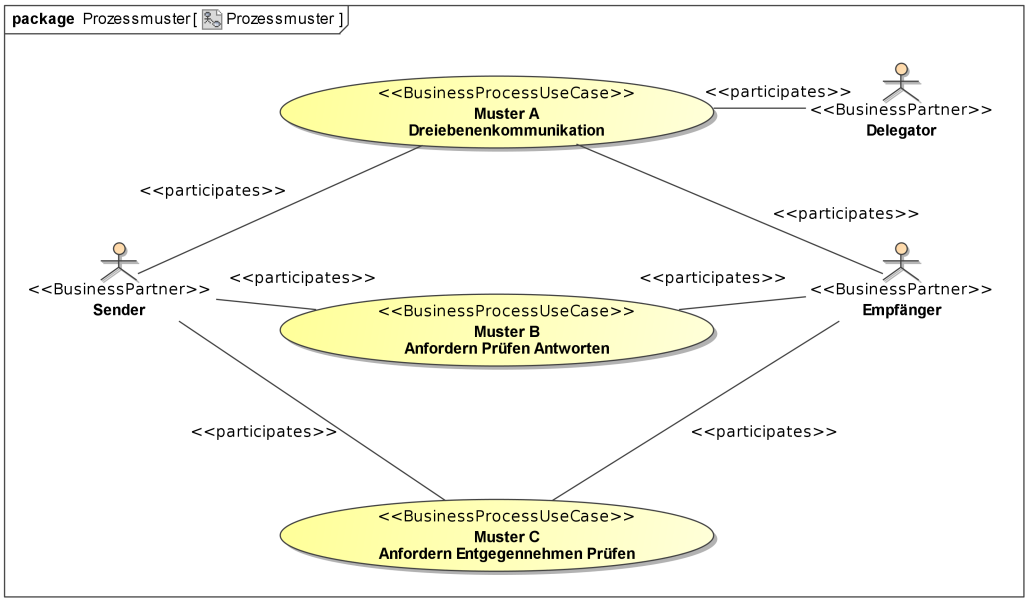


Abbildung B.8.: Use-Case-Diagramm „Prozessmuster“

Prozessmuster A: Dreiebenenkommunikation
Dieses Prozessmuster (Abb. B.9) ist eine Erweiterung des Sender-Empfänger-Schemas, bei dem ein Delegator oder Makler zwischen Sender und Empfänger geschaltet ist. In den untersuchten Prozessen ist dies in der Regel dann der Fall,

- wenn dem Sender die Rechtsgrundlage zur Anforderung von Informationen fehlt und an dessen Stelle der Delegator tritt, der diese Rechtsgrundlage hat. In diesem Fall weichen die Strukturen der Nachrichten zwischen Sender und Delegator sowie zwischen Delegator und Empfänger von einander ab.
- Oder wenn eine formal zuständige Stelle die Facharbeit an eine andere Behörde delegiert hat und eine Nachricht daher durchreicht. In diesem Fall weichen die Strukturen der Nachrichten zwischen Sender und Delegator sowie zwischen Delegator und Empfänger nicht von einander ab.

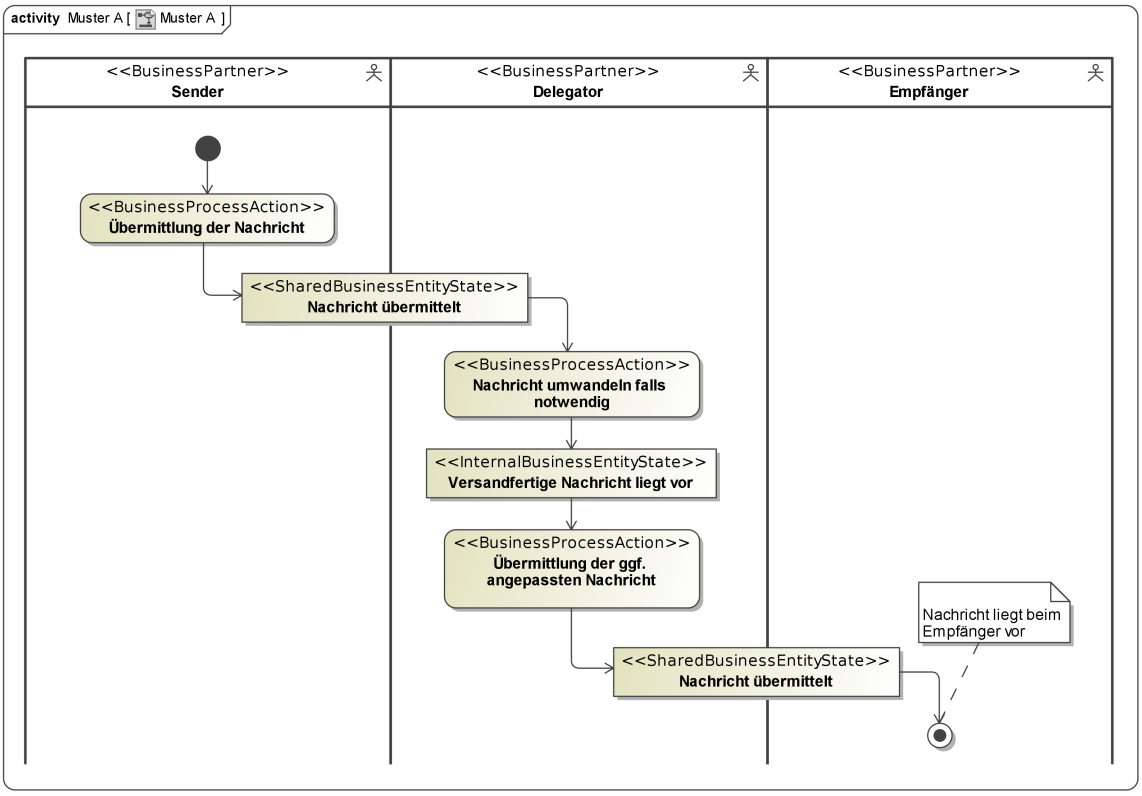


Abbildung B.9.: Aktivitätsdiagramm „Muster A: Dreiebenenkommunikation”

Prozessmuster B: Anfordern, prüfen, antworten
Dieses Prozessmuster (Abb. B.10) ist eine Erweiterung des Sender-Empfänger-Schemas, bei dem nach der Übermittlung der Nachricht (Daten) eine Prüfung der übermittelten Daten durchgeführt wird. Zeigt die Prüfung Fehler auf, wird eine Korrekturanforderung (Fehlermeldung) generiert und an den ursprünglichen Sender der Daten übermittelt. In den untersuchten Prozessen wird die Prüfung regelmäßig nicht während der Übermittlung durchgeführt, sondern erst nach Eingang der Nachricht bzw. der Daten. Treten Fehler auf, die bereits bei der Übermittlung hätten detektiert werden können, führt dies zu unnötiger Kommunikation und unnötigen Prüfungen auf Seite des Empfängers. Eine Optimierung dieses Prozessmusters (Abb. B.11) bindet einen P23R in die Kommunikation ein, der während der Übermittlung der Nachricht (Daten) gemäß P23RPrinzip eine Prüfung der zu übermittelnden Daten durchführt. Verläuft die Prüfung erfolgreich, werden die Daten an den Empfänger übermittelt und es wird eine Erfolgsmeldung generiert und an den ursprünglichen Sender der Daten übermittelt. Zeigt die Prüfung Fehler auf, wird eine Korrekturanforderung generiert und an den ursprünglichen Sender der Daten übermittelt.
Prozessmuster B basiert auf der Annahme, dass ein signifikanter Teil der Prüfungen bereits während der Übermittlung auf den zu übertragenden Daten ausgeführt werden kann.

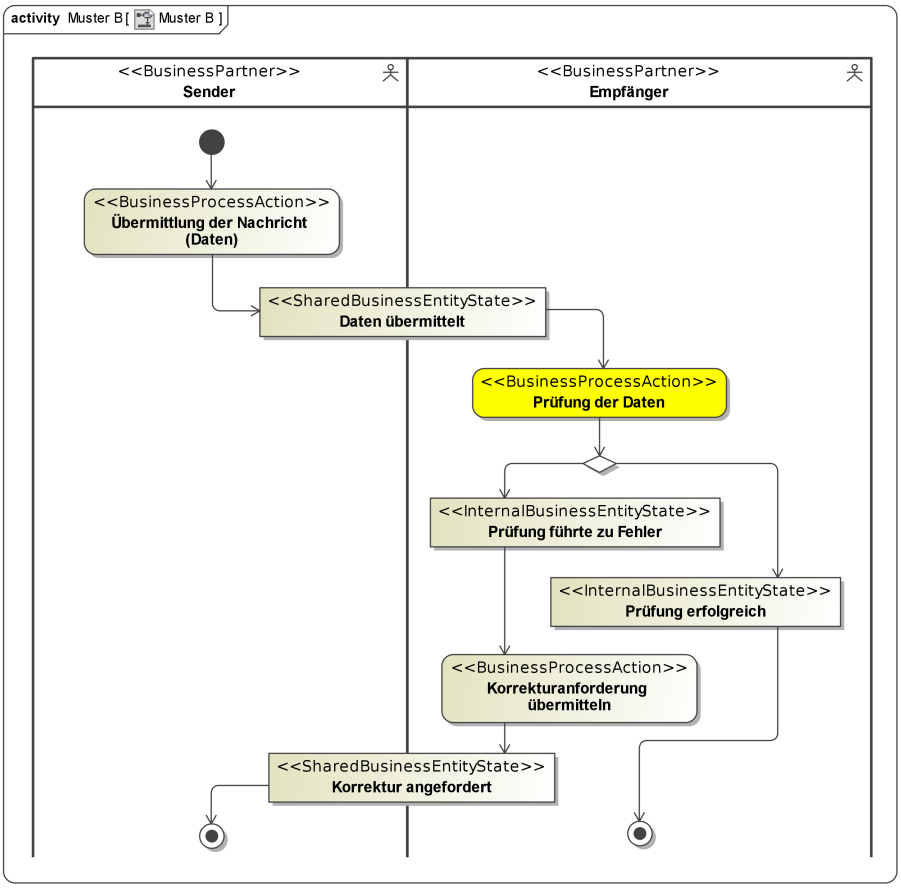


Abbildung B.10.: Aktivitätsdiagramm „Muster B: Übermitteln, prüfen, antworten”

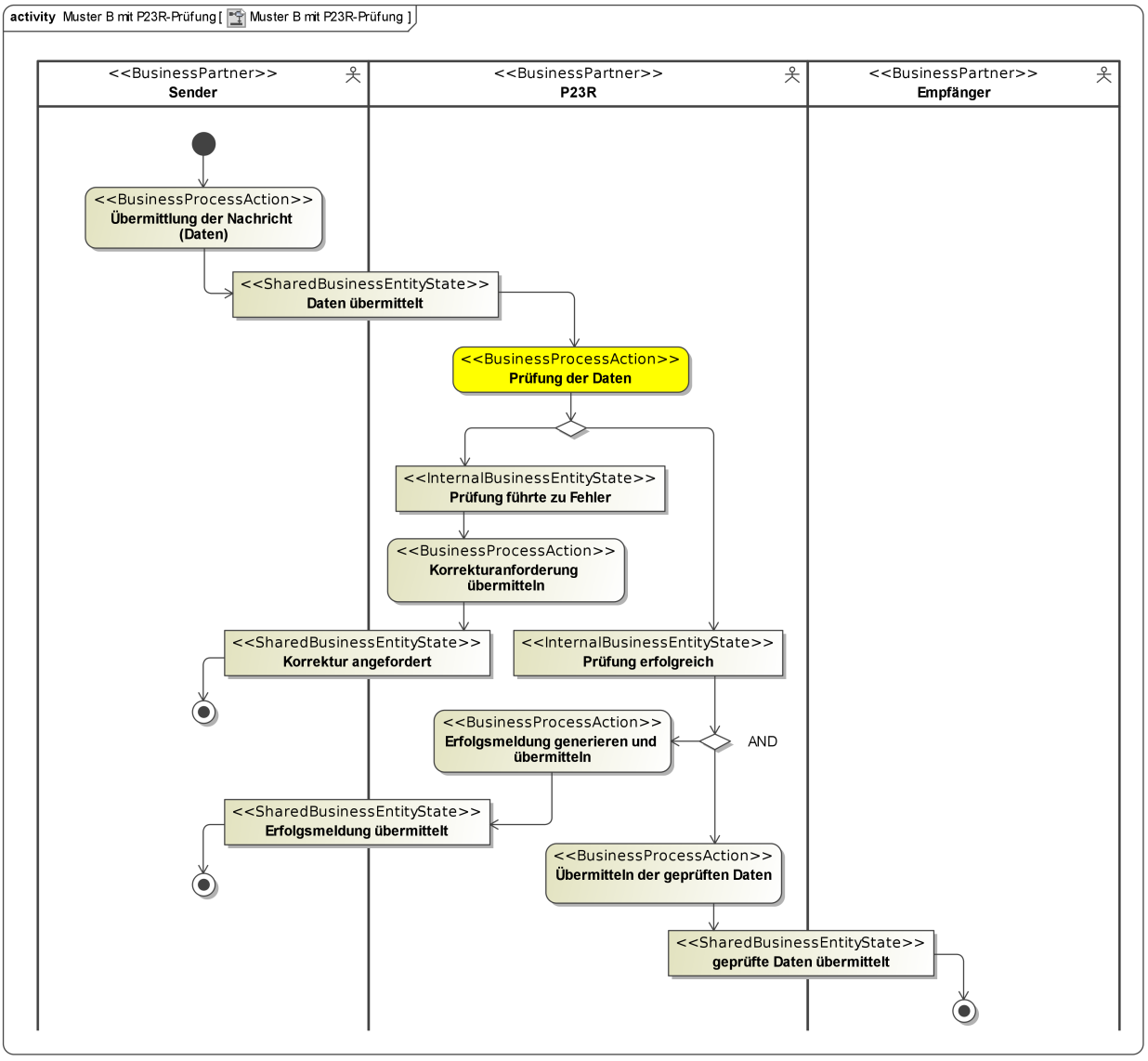


Abbildung B.11.: Aktivitätsdiagramm „Muster B mit P23R-Prüfung optimiert: Übermitteln, P23r prüft und antwortet”

Prozessmuster C: Anfordern, entgegennehmen, prüfen

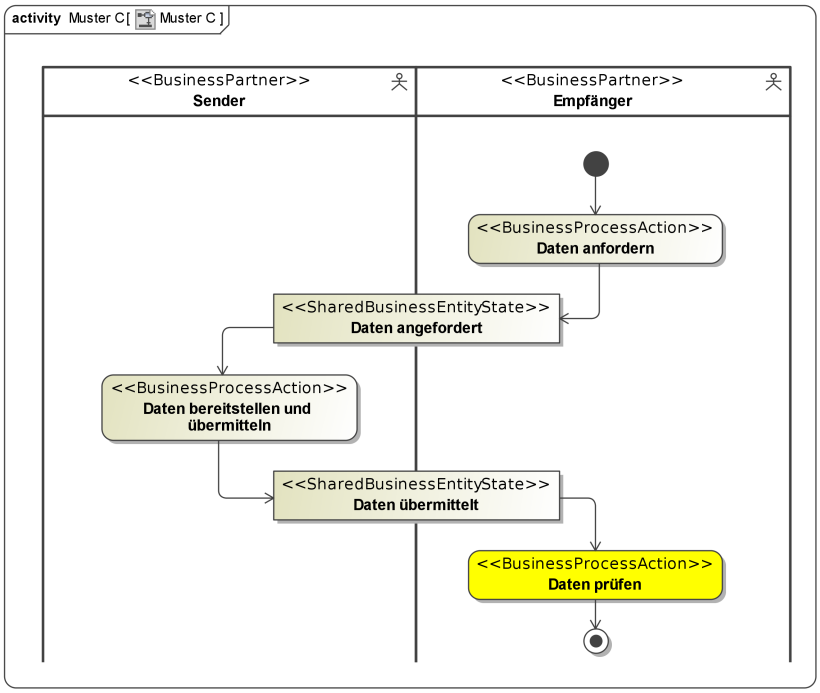


Abbildung B.12.: Aktivitätsdiagramm „Muster C: Anfordern, entgegennehmen, prüfen”

Das Prozessmuster C (Abb. B.12) tritt auf, wenn Daten zunächst angefordert und dann vom Sender an den Empfänger übermittelt werden, bei der Übermittlung der Daten keine oder nur einfache Prüfungen (z. B. Konformität zu einem XML-Schema) ausgeführt werden können und erst beim Empfänger umfangreiche Prüfungen der Daten durchgeführt werden.
Dies ist beispielsweise dann der Fall, wenn nur Datenausschnitte (z. B. Korrekturen einzelner Datensätze) übertragen werden, deren Prüfung erst beim Empfänger unter Hinzunahme weiterer Informationen sinnvoll vorgenommen werden kann. Solche weiteren Informationen sind z. B. ältere Informationen zu einem übermittelten Objekt (Zeitreihenuntersuchung) oder Informationen aus mit dem übermittelten Objekt verknüpften anderen Objekten. Ein Beispiel wäre die Übermittlung von Daten einer Kläranlage, die beim Datenempfänger mit den Daten eines verknüpften Siedlungsgebiets verglichen werden. Diese nachgelagerten Prüfungen sind meist aus fachlichen, oft auch aus rechtlichen Gründen von spezialisiertem Personal durchzuführen.
Die Anforderung der Daten und deren Übermittlung war gemäß dem P23R-Prinzip realisierbar, wohingegen die nachgelagerte Prüfung der Daten nicht durch das P23R-Prinzip abgedeckt wurde.
Module für zukünftige P23R Regelimplementationen
Die zukünftige Entwicklung einer größeren Zahl produktionstauglicher P23R-Regeln wird Aufwände nach sich ziehen, die sich durch den Einsatz dokumentierter und wiederverwendbarer P23R-Regelkomponenten senken lassen. Die Untersuchung der Berichtsprozesse für die Erfüllung der Kommunalabwasserrichtlinie 91/271EWG und

der Prozesse im Zusammenhang mit der Selbstüberwachung von Abwasseranlagen gemäß §61 Wasserhaushaltsgesetz (WHG) hat die oben dargelegten Prozessmuster A, B und C als wiederkehrende und über die triviale Übermittlung Sender-Empfänger hinausgehende Teile der Berichtsprozesse identifiziert. Für die zukünftige Entwicklung dokumentierter und wiederverwendbarer P23RRegelkomponenten wird empfohlen, zunächst die regelmäßig benötigten, trivialen Operationen • entgegennehmen einer Nachricht • auslösen einer Nachricht (Trigger) • senden einer Nachricht • umwandeln einer Nachricht und nachfolgend die Prozessmuster A, B und C zu unterstützen.

C. Prozess EU-Kommunalabwasser-Richtlinie (91/271/EWG)

Die folgende Kapitel C ist dem Abschlussbericht der ENDA KG entnommen. Die Regel steht ab Juli 2014 zum Download unter http://p23r.enda.eu bereit.
Die Prozessaufbereitung diente der Vorbereitung der Umsetzung des elektronischen Nachrichtenaustauschs nach dem P23R-Prinzip. Ziel war die Beschreibung der Kommunikationsprozesse für die EU-Richtlinie 91/271/EWG Kommunalabwasser und die Selbstüberwachung von Abwasseranlagen gemäß §61 Wasserhaushaltsgesetz (WHG) in einer etablierten Notation. Weiteres Ziel war es, zu prüfen, ob eine Umgestaltung der Prozesse sinnvoll ist, um die Realisierung einer Kommunikation nach dem P23R-Prinzip zu vereinfachen. Mit Blick auf die Automatisierung zahlreicher weiterer Kommunikationsprozesse der betrieblichen Umweltberichterstattung sollten Module für zukünftige P23R Regelimplementationen definiert und in Form von Umweltprozessmustern modelliert werden. Weiteres Ziel war es, die Qualitätssicherungsanforderungen (QS-Anforderungen) zu untersuchen und zu kategorisieren, um Module für zukünftige P23R Regelimplementationen zu definieren.

C.1. Prozessmodellierung

C.1.1. Prozessmodellierung für das Bundesland Bayern

Die Berichterstattung gemäß Kommunalabwasserrichtlinie wird im Use-Case-Diagramm „Kommunalabwasserberichterstattung 91/271/EWG“ (Abb. C.1) und im gleichnamigen Aktivitätsdiagramm (Abb. C.2) beschrieben. Die Aktivität „Kommunalabwasserberichterstattung 91/271/EWG“ greift dabei wiederum auf die Aktivität „Kläranlagenjahresbericht erstellen“ (Abb. C.3) zurück. Die im Aktivitätsdiagramm „Kommunalabwasserberichterstattung 91/271/EWG“ Bayern (Abb. C.2) dargestellte und vom Landesamt für Umwelt durchgeführte Prüfung umfasst alle Qualitätssicherungs-Prüfungen, die auf der zentralen Anwendung „e-Kommunalabwasser“ des Umweltbundesamts durchgeführt werden.

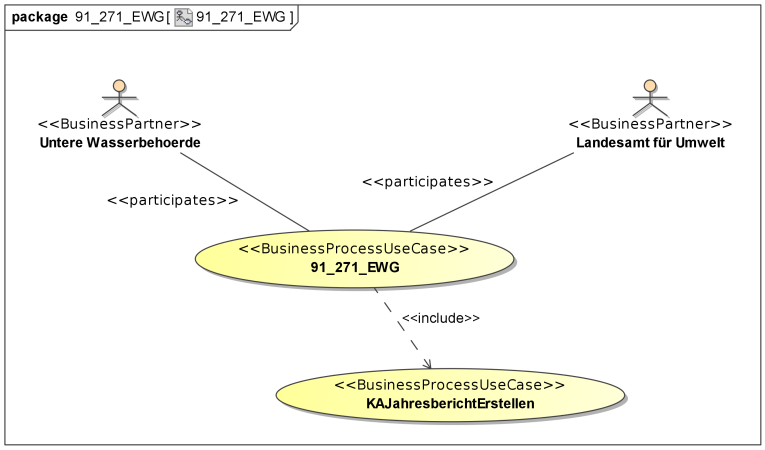


Abbildung C.1.: Use-Case-Diagramm: „Kommunalabwasserberichterstattung 91/271/EWG”

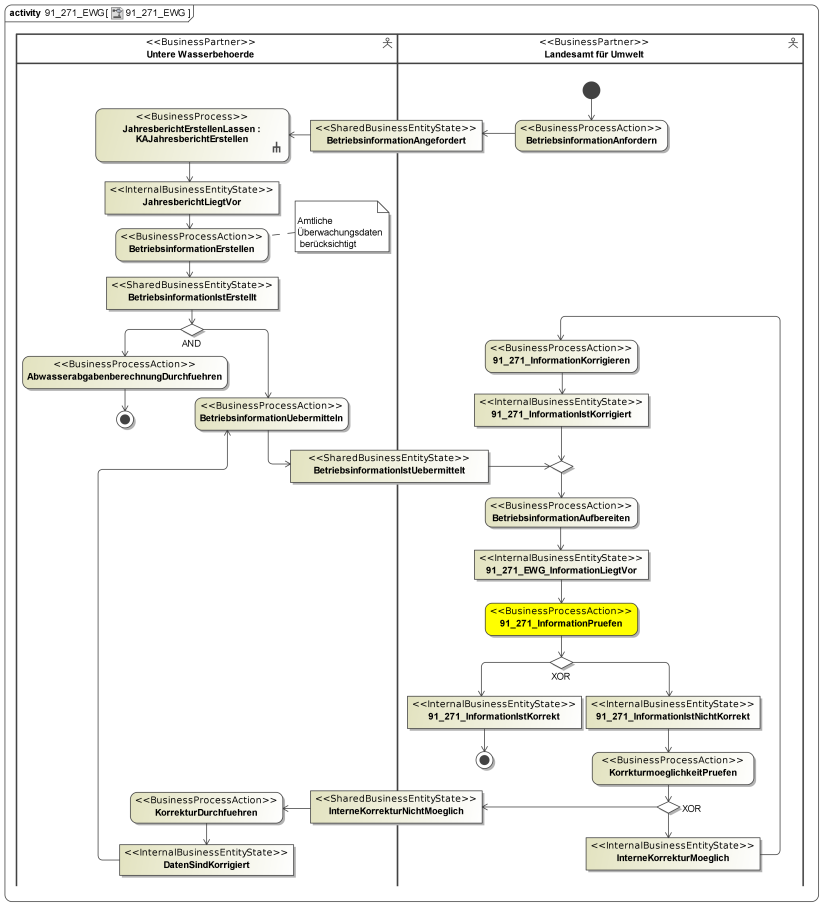


Abbildung C.2.: Aktivitätsdiagramm: „Kommunalabwasserberichterstattung 91/271/EWG”

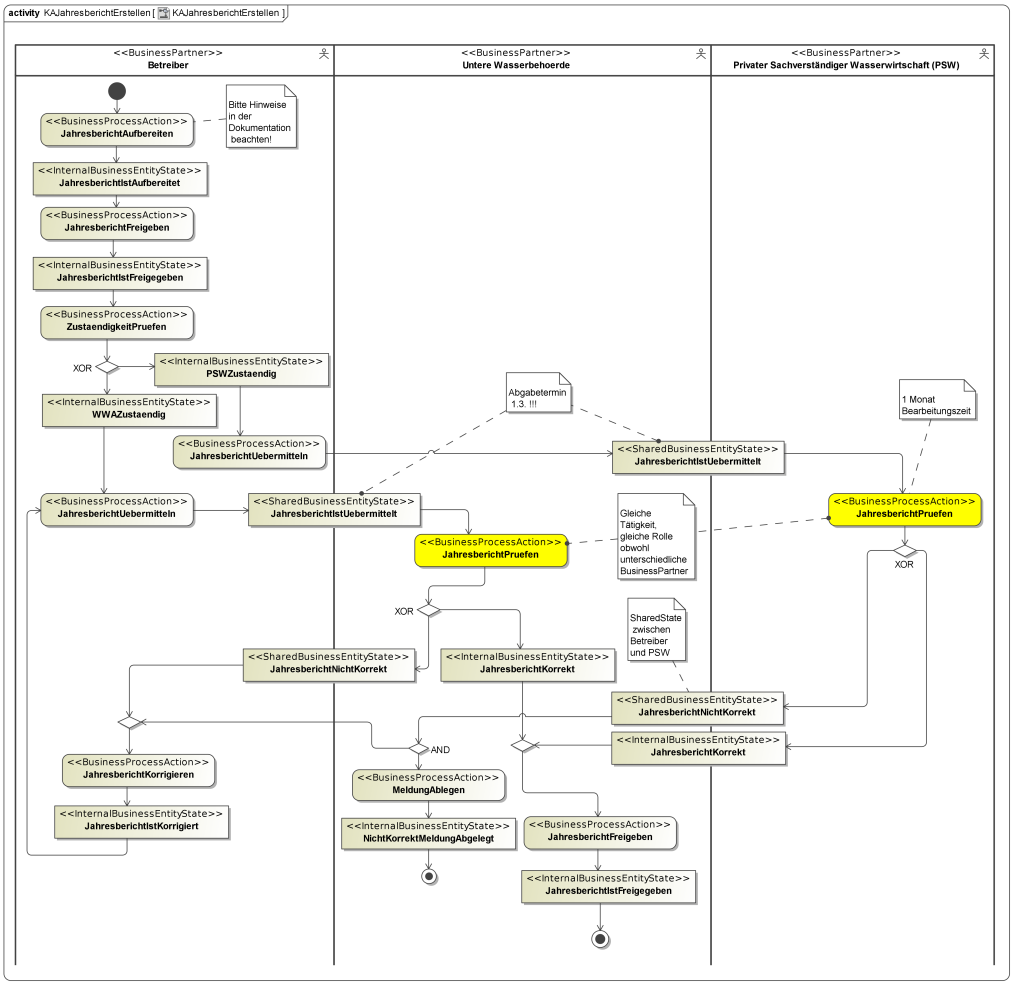


Abbildung C.3.: Aktivitätsdiagramm: „Kläranlagenbericht erstellen”

C.1.2. Prozessmodellierung für das Bundesland Sachsen

Die Anforderungen der EU-Richtlinie 91/271/EWG werden auf Basis der Sächsischen Kommunalabwasserverordnung umgesetzt. Für die Berichterstattung gemäß Kommunalabwasserrichtlinie 91/271/EWG fertigt in Sachsen das Umweltministerium jedes Mal einen Erlass.
Die Aktivität „Ermittlung Jahreswerte und Frachten EU-Richtlinie 91/271/EWG“ (Abb. C.4) zeigt den Prozess für die Ermittlung der Daten der Kommunalabwasserberichterstattung, der in einem Prozessschritt auf eine weitere Aktivität „Eintragen der Frachtwerte“ (Abb. C.5) zurückgreift. Neben den Berichtspflichten gemäß §61 WHG sind die Kläranlagenbetreiber bzw. Abwasserbeseitigungspflichtige (ABP) gehalten, vor allem folgenden Institutionen bzw. Kampagnen zu berichten:

- ABP berichten dem statistischen Landesamt auf Grundlage des Umweltstatistikgesetzes (folglich verpflichtend)
- Betreiber berichten dem Fachverband DWA (Deutsche Vereinigung für Wasserwirtschaft; Abwasserfachverband), der regelmäßig Leistungsvergleiche durch-

führt, an denen jeweils ca. 80-90% der Betreiber teilnehmen (auf freiwilliger Basis)

- Betreiber berichten dem Abwasser-Benchmarking, das vor allem den Energieverbrauch der Anlagen untereinander zu vergleichen sucht (auf freiwilliger Basis).

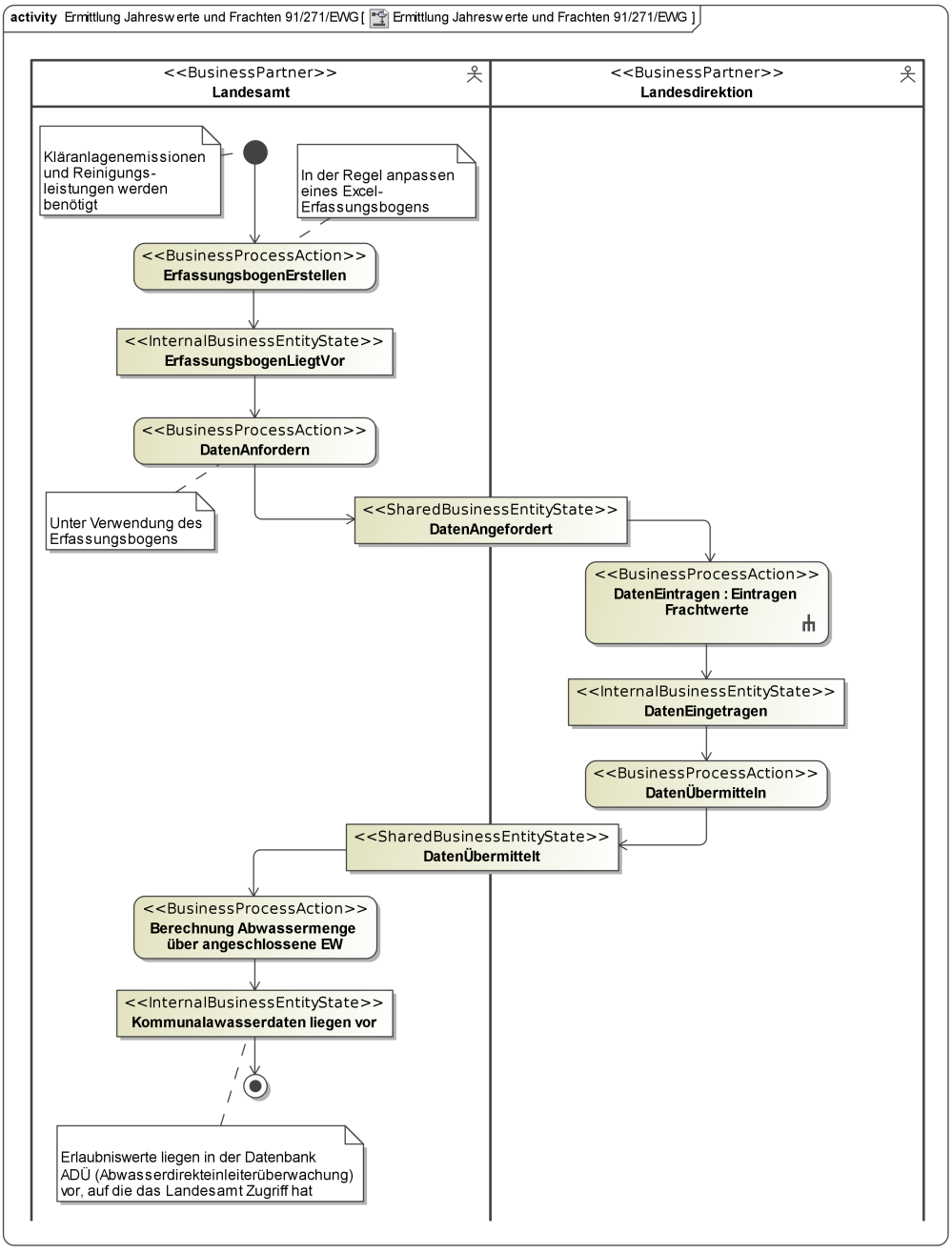


Abbildung C.4.: Aktivitätsdiagramm „Ermittlung Jahreswerte und Frachten EU-Richtlinie 91/271/EWG“ Sachsen

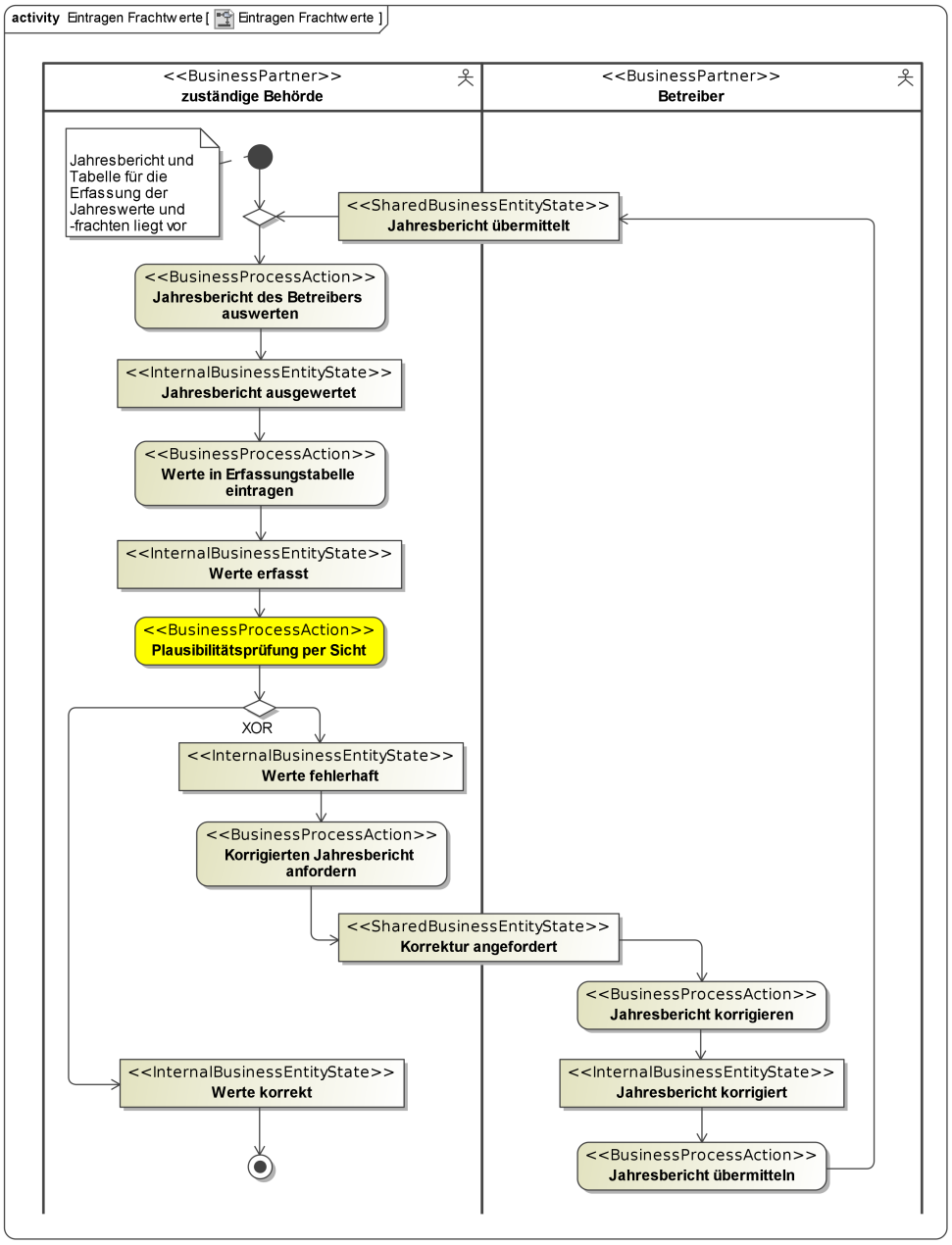


Abbildung C.5.: Aktivitätsdiagramm „Eintragen der Frachtwerte“ Sachsen

C.1.3. Prozessmodellierung für den Bund

Die hier dargestellten Diagramme:

- Use-Case-Diagramm „91/271/EWG“ Bund (Abb. C.6)
- Aktivitätsdiagramm „Anforderung EU“ Bund (Abb. C.7)
- Aktivitätsdiagramm „Anforderung Bund“ (Abb. C.8)
- Aktivitätsdiagramm „Korrekturanforderung Bund“ (Abb. C.9)

entstammen der Modellierung im Projekt XUBetrieb und wurden für die Darstellung an dieser Stelle leicht grafisch und inhaltlich überarbeitet. Sie sind hier aufgeführt, da sie für die Ableitung von Umwelt-Prozessmustern ebenso als Ausgangsbasis dienen, wie die in diesem Projekt neu erhobenen Prozesse auf kommunaler Ebene und Landesebene der Länder Bayern und Sachsen.

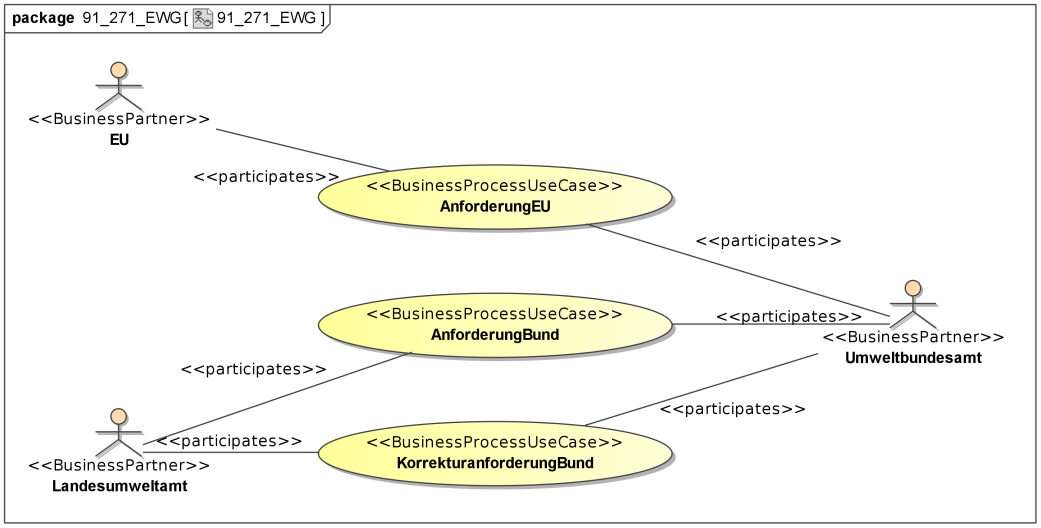


Abbildung C.6.: Use-Case-Diagramm „91/271/EWG“ Bund

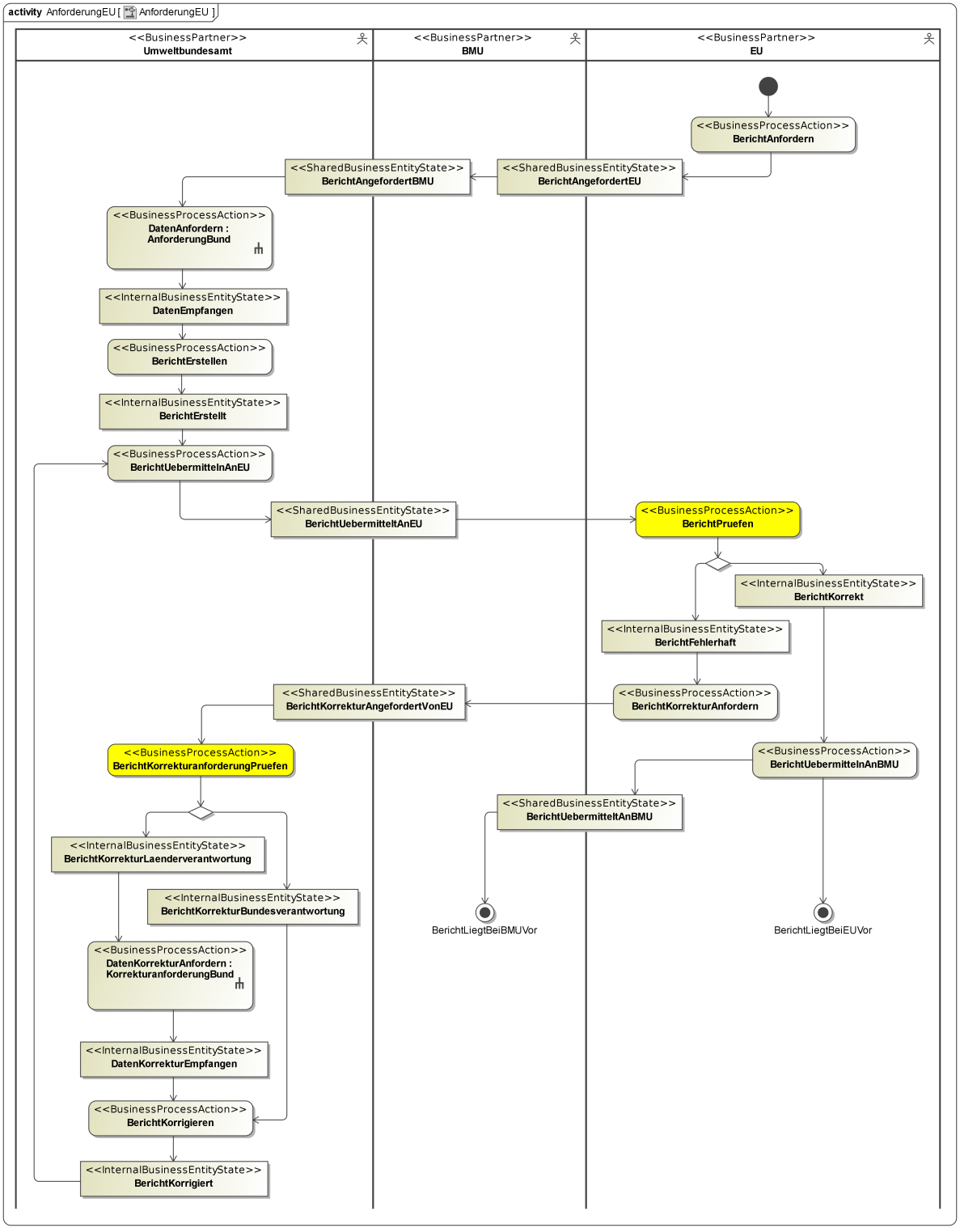


Abbildung C.7.: Aktivitätsdiagramm „Anforderung EU“ Bund

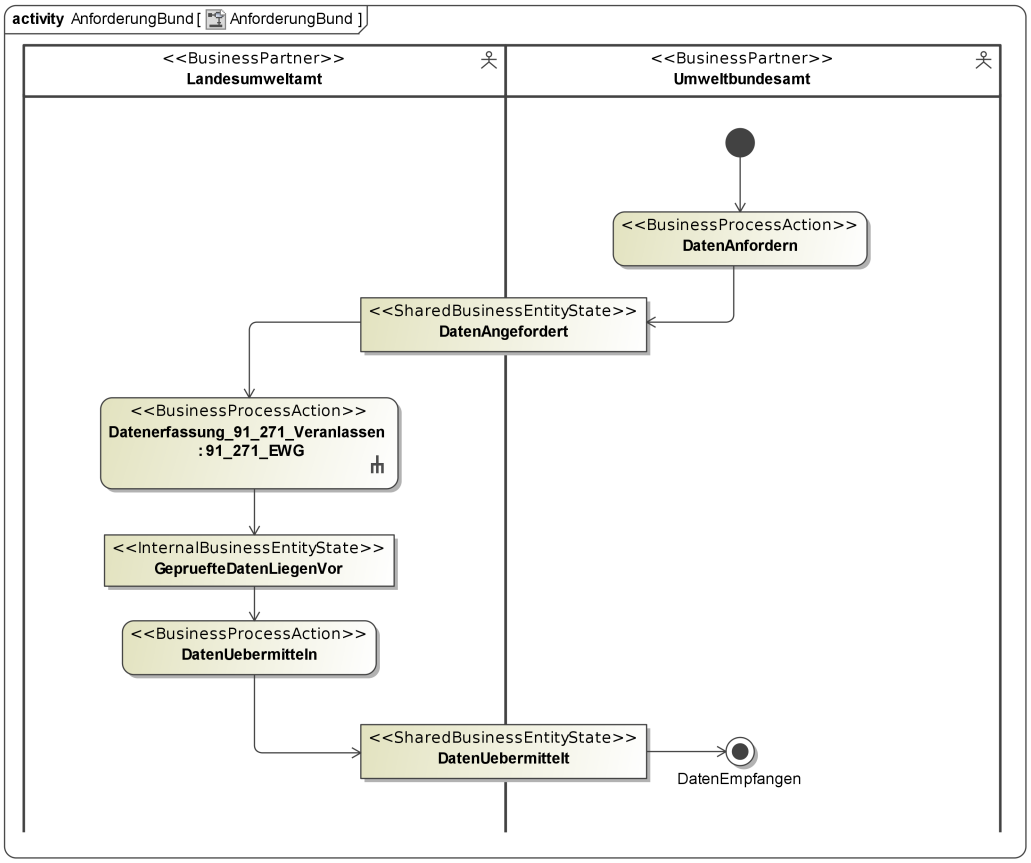


Abbildung C.8.: Aktivitätsdiagramm „Anforderung Bund“

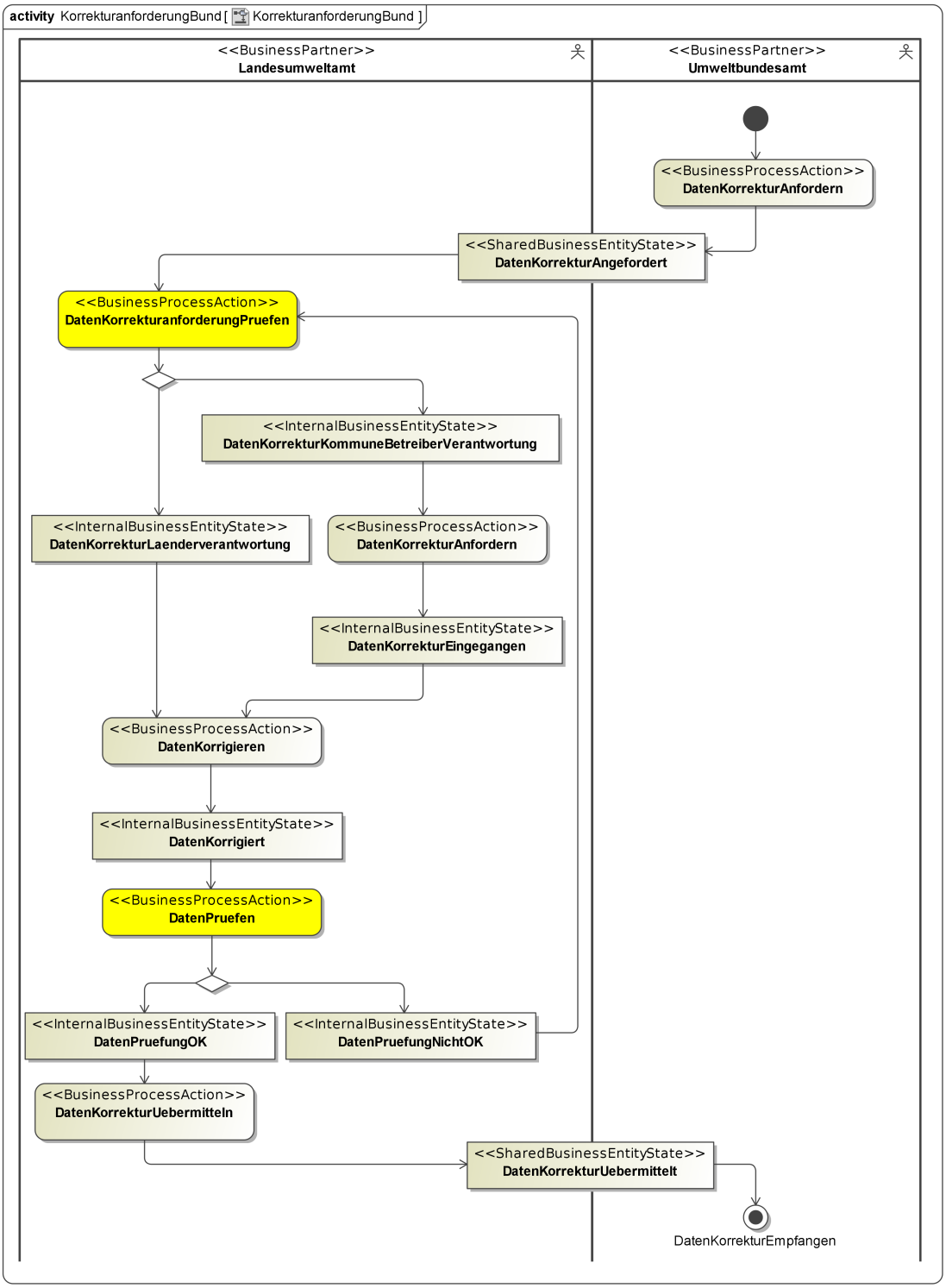


Abbildung C.9.: Aktivitätsdiagramm „Korrekturanforderung Bund“

C.2. Detaillierte Beschreibung der QS-Prüfungen bei Übermittlung von Kommunalabwasserberichtsdaten

C.2.1. Kategorisierung von QS-Prüfungen

Für die Unterstützung der P23R Regelkomponenten-Entwickler bei der Implementation von dokumentierten und wiederverwendbaren P23R-QS-Regelkomponenten war es zunächst notwendig, eine Strukturierung der verschiedenen Arten von Prüfungen zu erreichen. Nur so konnten funktional getrennte, sich ergänzende Bausteine entworfen werden, aus denen sich später P23R-QS-Regeln zusammensetzen lassen.
Zunächst waren die Hauptkategorien[1] strukturelle Anforderungen, inhaltliche Anforderungen und Plausibilitätsanforderungen von einander zu unterscheiden. Diese Unterteilung war jedoch für eine Umsetzung in P23R-QS-Regelkomponenten noch zu grob: Ein Teil der strukturellen Anforderungen konnte durch eine XML Schema-Validierung geprüft werden, andere strukturelle Anforderungen bzgl. Wertebereichen und Multiplizitäten ließen sich praktisch nicht sinnvoll durch eine XML Schema-Validierung überprüfen. Plausibilitätsanforderungen sind vielschichtig. Ihre Prüfungen haben häufig unterschiedliche Anforderungen an den notwendigen Datenumfang. Daher wurde hier eine feinere Unterteilung der QS-Prüfungen vorgenommen, die sich einerseits an den Erfordernissen des P23R-Prinzips (z. B. Unterteilung der Nachrichten in einen Meta- und einen Payload-Teil) und andererseits an den Prüfungen der Berichtsdaten der Kommunalabwasserberichterstattung orientierte:

- Versionsprüfungen der Meta- und der Payload-Daten
- XML Schema-Validierungen
- Strukturelle Prüfungen, Teil 1: Abhängige Pflichtfelder, Listenwerte aus Listen, deren Umfang den Einsatz von XML Schema zur Validierung ausschließt
- Strukturelle Prüfungen, Teil 2: Multiplizitäten unter Berücksichtigung der Objektstatus
- Inhaltliche Prüfungen: Umfang der berichteten Objekte; benötigt Zugriff auf Daten der Vorperiode
- Plausibilität objektintern
- Plausibilität objektübergreifend
- Plausibilität im zeitlichen Verlauf: Zeitreihenuntersuchung; benötigt Zugriff auf Daten möglichst vieler Vorperioden

C.2.2. Detaillierte Beschreibung der QS-Anforderungen: Notation und Allgemeines

Die folgenden Bedingungen, die für einen gültigen Bericht gelten müssen, sind jeweils verbal beschrieben. Eine formale und eindeutige Darstellung in der Sprache

[1] Lüttgert, M. ; Senf, Ch.: Abschlussbericht für das Projekt XML Repository Betriebliche Stamm- und Berichtsdaten als Teil des XÖV des Bundes (D2- 06-4) — XUBetrieb — Phase III / ENDA GmbH & Co. KG, KRALLMANN AG. 2011. – Forschungsbericht

OCL kann dem Originaldokument des Auftragnehmers ENDA entnommen werden. Dies diente als Grundlage für eine Programmierung der QS-Anforderungen. Objekte und – wo angebracht – auch ihre Kindelemente wurden durch einen XPath-Ausdruck im XUKommunalabwasser-Schema identifiziert. Damit wurde der Aufwand für die Implementierung der Prüfungen minimiert, da die Daten in der Struktur genau jenes XUKommunalabwasser-Schemas vorliegen müssen.
Die Prüfungen der Version sind erst bei Einsatz des P23R-Prinzips erfolgt. Die XML Schema-Validierungen waren trivial und daher nicht näher zu dokumentieren. Zeitreihenuntersuchungen waren nicht Bestandteil der Kommunalabwasserberichtsprüfungen. Daher werden diese drei Arten von QS-Prüfungen im folgenden Text nicht detailliert.

C.2.3. Prozessschritt Strukturelle Prüfungen, Teil 1: Abhängige Pflichtfelder und lange Listen

Hier werde die Existenz von Feldinhalten geprüft, die nur unter bestimmten Bedingungen verpflichtend zu übermitteln sind. Weiterhin werden diejenigen Listenwerte geprüft, deren Listenumfang eine XML Schemaprüfung praktisch verhindert.

C.2.3.1. Siedlungsgebiete

- Wenn sich die Schadstofffracht verändert hat, so muss eine dazugehörige Bemerkung angegeben werden
- Ist die Nominalbelastung größer 100.000 EW und der in IAS gesammelte Anteil größer gleich 2.000 EW, dann muss mindestens eine Behandlungsart angegeben werden
- Ist das Siedlungsgebiet Teil einer Großstadt, so ist diese anzugeben
- Werden Teile des Siedlungsabwassers noch nicht gesammelt, so muss ein Datum angegeben werden, ab dem dies behoben ist

C.2.3.2. Kläranlagen

Bedingungen zu Stickstoff

- Übersteigt die Ausbaugröße 10.000 EW, so muss mindestens ein Erlaubniswert – Maximalwert der Konzentration in mg/l (erlaubniswert) oder die prozentuale Mindestreduktion (erlaubniswertReduktion) – zu Stickstoff angegeben werden

Die Angaben zur Einhaltung sowohl der europäischen als auch der nationalen Anforderungen (einhaltungAnforderungEU, einhaltungAnforderungD) werden ab Version 1.1.5 des Schemas XUKommunalabwasser für alle Parameter (Stickstoff (N), Phosphor (P), CSB und BSB5) verpflichtend und müssen daher nicht mehr als separate Prüfung implementiert werden.
Bedingungen zu Phosphor

- Übersteigt die Ausbaugröße 10.000 EW, so muss mindestens ein Erlaubniswert – Maximalwert der Konzentra..on in mg/l (erlaubniswert) oder die prozentuale Mindestreduktion (erlaubniswertReduktion) – zu Phosphor angegeben werden

Weitere Bedingungen

- Unabhängig von der Ausbaugröße muss der Erlaubniswert zu CSB in mg/l oder Prozent Reduktion angegeben werden
- Unabhängig von der Ausbaugröße muss der Erlaubniswert für BSB5 in mg/l oder Prozent Reduktion angegeben werden
- Ist die Abwasserbehandlungsart einer Kläranlage weitergehend, so muss ein Verfahren angegeben werden

C.2.3.3. Einleitstellen

- Wird das Abwasser in den Boden (Land) eingeleitet (verrieselt), so ist die Nutzung (auch: „Gebrauch“) anzugeben, sonst das Einleitgewässer (inkl. Name, Art, Einzugsgebiet etc.)
- Wird das Abwasser in den Boden (Land) eingeleitet (verrieselt), so ist der Grundwasserkörper anzugeben, sonst der Wasserkörper (gemäß WRRL)
- Verwendete Wasser- und Grundwasserkörper müssen in der jeweiligen externen Liste (beide sind nicht im Schema hinterlegt) enthalten sein

C.2.4. Prozessschritt Strukturelle Prüfungen, Teil 2: Multiplizität und Objektstatus

C.2.4.1. Kardinalitäten Siedlungsgebiet-Kläranlage

- Jedem aktiven Siedlungsgebiet muss mindestens eine aktive Kläranlage zugewiesen werden und jeder aktiven Kläranlage mindestens ein aktives Siedlungsgebiet

C.2.4.2. Kardinalitäten Kläranlage-Einleitstelle

- Jeder aktiven Kläranlage muss mindestens eine aktive Einleitstelle zugeordnet sein

C.2.5. Prozessschritt Inhaltliche Prüfungen: Bezug zur Vorperiode

Für außerhalb Deutschlands gelegene Objekte ist Deutschland nicht in der Verantwortung, die historische Entwicklung nachzuvollziehen. Eine Prüfung auf Konsistenz mit Meldungen in der vorherigen Berichtsperiode wird daher für außerhalb Deutschlands gelegene Objekte nicht vorgenommen.

C.2.5.1. Nicht mehr vorhandene Objekte

Alle in der vorherigen Berichtsperiode als ak.tiv gemeldeten Siedlungsgebiete, Kläranlagen und Einleitstellen innerhalb Deutschlands müssen auch in der aktuellen Berichtsperiode gemeldet werden (Wiederholungsmeldung).

C.2.5.2. Bereits als inaktiv gemeldete Objekte

In der Vorperiode bereits als inaktiv gemeldete Objekte (Siedlungsgebiete, Kläranlagen und Einleitstellen), jeweils identifiziert durch das Element „id“, dürfen in der aktiven Berichtsperiode nicht wieder gemeldet werden (Exzessmeldung)

C.2.5.3. Inaktive Objekte, die in der Vorperiode nicht berichtet wurden

Als inaktiv berichtete Siedlungsgebiete, Kläranlagen und Einleitstellen, jeweils identifiziert durch das Element „id“, müssen bereits in der Vorperiode als aktiv berichtet worden sein (Abmeldung)

C.2.6. Prozessschritt Plausibilitätsprüfung Anschlussgrade Objektintern

C.2.6.1. Unstimmigkeiten der Anschlussgrade innerhalb des Siedlungsgebiets – Summe der Hauptfrachten

Für Siedlungsgebiete ab 2.000 EW muss die Summe der Anteile Hauptfrachtströme (in Kanalisation, IAS und nicht gesammeltes Abwasser) zusammen 100% ergeben. Eine exakte Gleichheit der numerischen Größen ist in der Praxis oft nicht zu erreichen, weshalb eine Toleranz von 0,1% vorzusehen ist

C.2.6.2. Unstimmigkeiten der angeschlossenen Frachten – Kanalisationsbilanz

Für Siedlungsgebiete ab 2.000 EW müssen die Teilstromfrachten der Kanalisation zusammen 100% ergeben. Eine exakte Gleichheit der numerischen Größen ist in der Praxis oft nicht zu erreichen, weshalb eine Toleranz von 0,1% vorzusehen ist

C.2.6.3. Unstimmigkeiten der individuell gesammelten Frachten – IASBilanz

Für Siedlungsgebiete ab 2.000 EW darf der Anteil des per LKW in die Kläranlagen transportierten Abwassers den insgesamt in IAS gesammelten Anteil nicht übersteigen. Es ist eine Toleranz von 0,1% vorzusehen

C.2.6.4. Unstimmigkeiten der Anschlussgrade innerhalb der weiteren Behandlung von Schmutzfracht aus IAS

Für Siedlungsgebiete mit mehr als 100.000 EW Belastung und ab 2.000 EW in IAS muss die Summe der prozentualen IAS-Anteile, die Erstbehandlung, Zweitbehandlung (Stufe) und weitergehende Behandlung erfahren, gleich 100% sein. Eine Toleranz von 0,1% ist vorzusehen

C.2.6.5. Anlagen mit Unstimmigkeiten der Nährstofffrachten an den Einleitstellen

Die Nährstofffrachten des Ablaufs müssen geringer sein, als die des Zulaufs. Nährstoffe sind sowohl Stickstoff (Codewert: N) als auch Phosphor (Codewert: P). Eine Toleranz von 0,1% ist vorzusehen. Die optionale Einheit „unitCode“ kann ab Version 1.1.0 des XUKommunalabwasser Schemas nur noch den Wert kg/a (Codewert: KGA) annehmen.

C.2.7. Prozessschritt Plausibilitätsprüfung Anschlussgrade Objektübergreifend

C.2.7.1. Unstimmigkeiten zwischen der Nominalbelastung einer Anlage und der Summe der Nominalbelastungen der der Anlage zugeordneten Siedlungsgebiete

Die Summe der Belastungsanteile von Siedlungsgebieten, die in eine Kläranlage entwässern, muss 110% der Nominalbelastung dieser Kläranlage unterschreiten.

C.2.7.2. Unstimmigkeiten zwischen der Ausbaugröße einer Anlage und der Summe der Nominalbelastungen der der Anlage zugeordneten Siedlungsgebiete

Diese Prüfung soll nur Warnungen, nicht Fehler erzeugen. Die Summe der Belastungsanteile von Siedlungsgebieten, die in eine Anlage entwässern, sollte 110% der Ausbaugröße dieser Anlage nicht übersteigen. Diese Prüfung ist der vorherigen Prüfung sehr ähnlich, nur werden hier die Siedlungsbelastungen mit der Ausbaugröße der Kläranlage und nicht mit ihrer Nominalbelastung verglichen. Daher gelten hier auch die Erläuterungen zu den Variablen und Selbstreferenzen entsprechend.

C.2.7.3. Unstimmigkeiten zwischen der Nominalbelastung eines Siedlungsgebiets und den zugeordneten Anlagen

Die Summe der Belastungsanteile der jeweiligen einem Siedlungsgebiet zugeordneten Kläranlagen muss dem „Anteil der erzeugten Belastung des Siedlungsgebiets, der in Kläranlagen behandelt wird“ entsprechen. Eine exakte Gleichheit der numerischen Größen ist in der Praxis oft nicht zu erreichen, weshalb eine Toleranz von 0,1% vorzusehen ist.

C.2.8. UML2-Darstellung der QS-Prüfungen

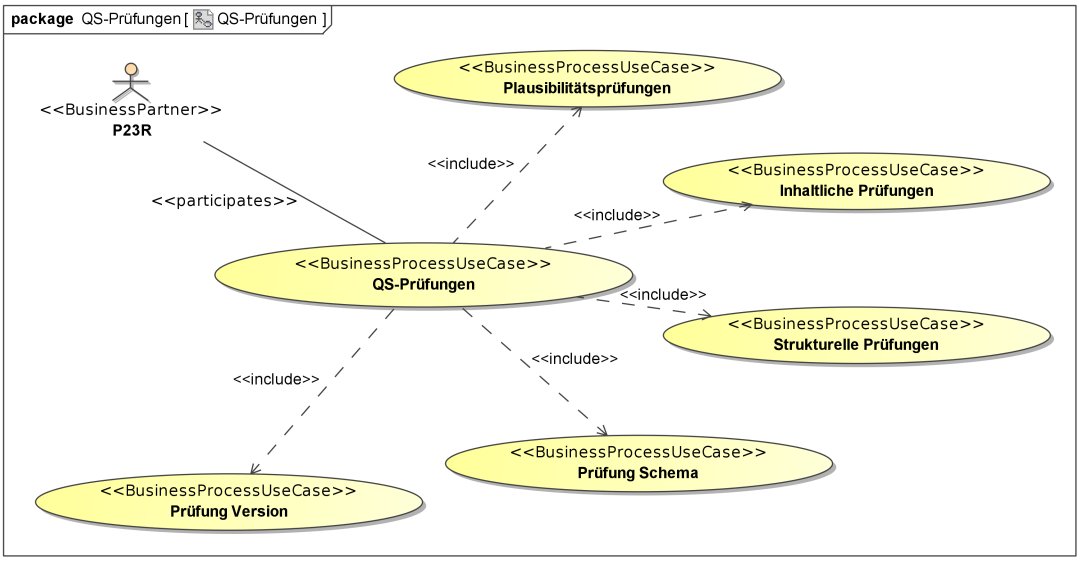


Abbildung C.10.: Use-Case-Diagramm „QS-Prüfungen“

Die Übersicht über die verschiedenen Prüfungsbestandteile der Qualitätssicherung ist im Use-Case-Diagramm „QS-Prüfungen“ (Abb. C.10) dargestellt. Der Prüfungsablauf auf oberster Ebene ist dem Aktivitätsdiagramm „QS-Prüfungen“ (Abb. C.11) zu entnehmen. Fehlgeschlagene Prüfungen der Version bzw. des Schemas, sowohl der Metadaten als auch der „Payload“-Daten führen zum Abbruch der Übertragung. Die Nachricht wird zurückgewiesen und ein entsprechendes Fehlerprotokoll (Fehlerlog) wird erstellt. Die in den Aktivitätsdiagrammen „Prüfung Version“ (Abb. C.12) und „Prüfung XML Schema“ (Abb. C.13) dargestellten Prüfungen werden als elementar eingestuft. Werden diese nicht erfolgreich durchlaufen, können die weiteren Prüfungen sich nicht darauf verlassen, die Daten an den durch die XML Schemata vereinbarten Stellen vorzufinden. Dann sind alle weiteren Prüfungen sinnlos und werden daher nicht durchgeführt. Die Aktivität „Log (Fehlerprotokoll) Schreiben“ (Abb. C.14) wird zwar von der Aktivität „QS-Prüfungen“ verwendet, ist jedoch nicht Bestandteil der eigentlichen QSPrüfungen, sondern eine Service-Funktion. Sie wird daher im Use-Case-Diagramm der QS-Prüfungen nicht mit aufgeführt.

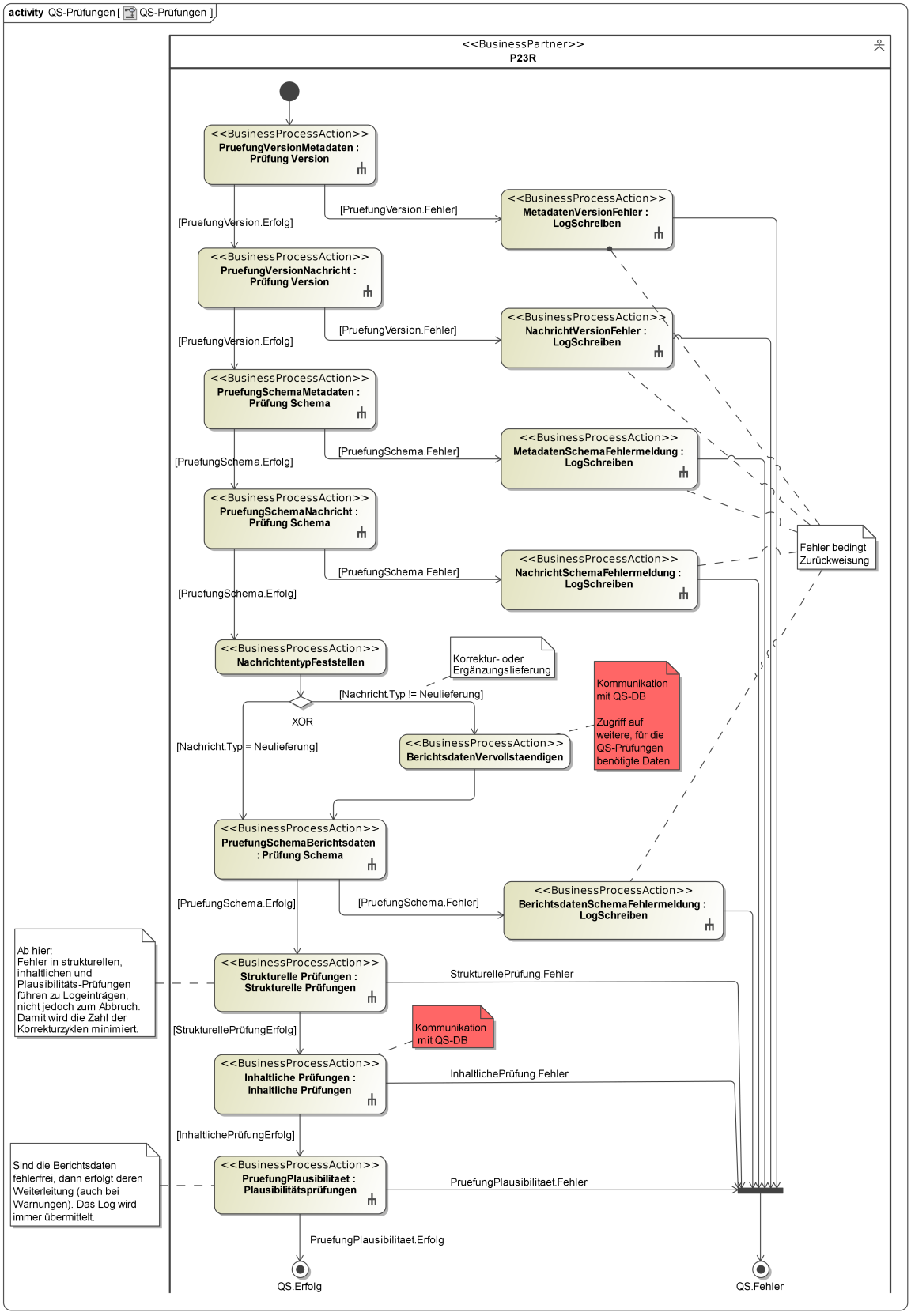


Abbildung C.11.: Aktivitätsdiagramm „QS-Prüfungen”

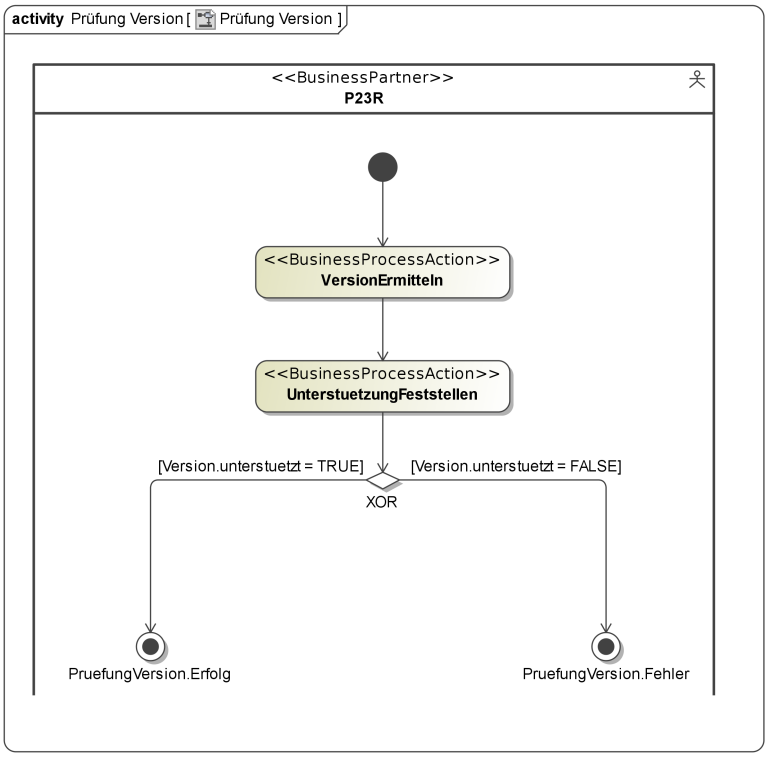


Abbildung C.12.: Aktivitätsdiagramm „Prüfung Version”

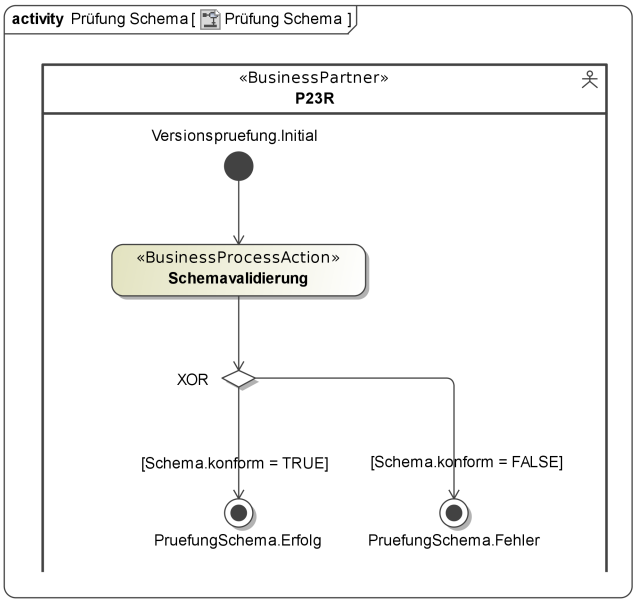


Abbildung C.13.: Aktivitätsdiagramm „Prüfung XML-Schema”

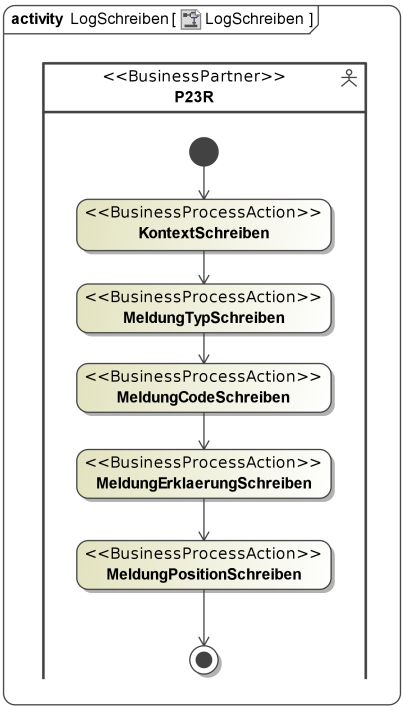


Abbildung C.14.: Aktivitätsdiagramm „Log (Fehlerprotokoll) Schreiben”

Erfüllt die Nachricht die (elementaren) Anforderungen an Version und Schema, werden alle strukturellen, inhaltlichen und Plausibilitätsprüfungen ausgeführt und protokolliert, auch wenn dabei Fehler auftreten. Zweck dieser Vorgehensweise ist, dass möglichst viele Fehler auf einmal detektiert, protokolliert und dem Sender der Berichtsdaten übergeben werden sollen. Damit liegt es in der Hand des Datenlieferanten, die Zahl der notwendigen Korrekturzyklen möglichst gering zu halten. Die im Aktivitätsdiagramm „Strukturelle Prüfungen“ (Abb. C.15) ausgeführten Prüfschritte brauchen für die Prüfung von Werten aus umfangreichen Listen Zugriff auf externe Daten. Diese sollten dem P23R über RESTful Services zur Verfügung gestellt werden.

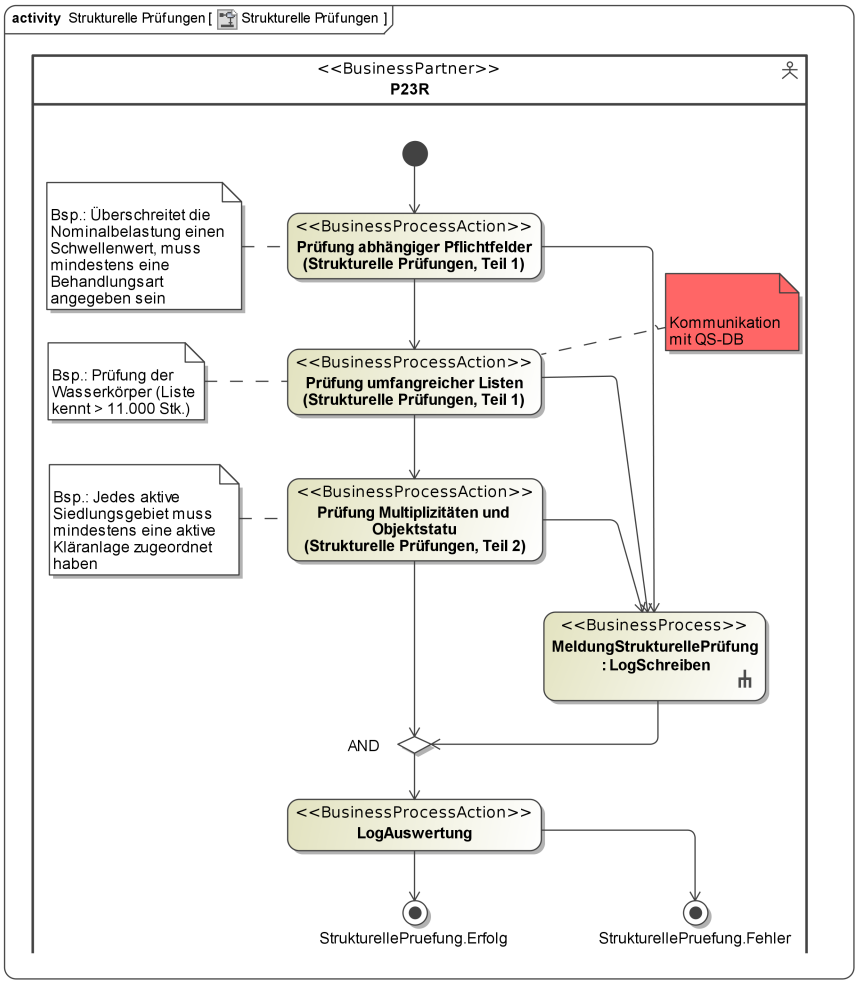


Abbildung C.15.: Aktivitätsdiagramm „Strukturelle Prüfungen”

Die drei Prüfschritte der Aktivität „Inhaltliche Prüfungen“ (Abb. C.16) benötigen Zugriff auf die Daten der vorherigen Berichtsperiode. Der P23R sollte auf diese Daten über RESTful Services zugreifen können.
Es kann nicht angenommen werden, dass die Datenlieferanten die Daten der vorherigen Berichtsperioden zu Prüfzwecken automatisiert mit übertragen können, da die überwiegende Zahl der Bundesländer die Berichtsdaten derzeit noch manuell zusammenstellt und nicht über historisch geführte, elektronische Kataster verfügt.
Von den im Aktivitätsdiagramm „Plausibilitätsprüfungen“ (Abb. C.17) abgebildeten Prüfschritten benötigt nur die Zeitverlaufsprüfung Zugriff auf die Daten (möglichst vieler) vorheriger Berichtsperioden. Für die Berichterstattung gemäß Kommunalabwasserrichtlinie sind solche Zeitverlaufsprüfungen nicht definiert. Dieser Prüfschritt ist hier der Vollständigkeit halber aufgeführt. Die QS-Prüfungen der PRTR-Berichterstattung ebenso wie die der Berichterstattung der Grundwasserzustandsdaten enthalten solche Zeitverlaufsprüfungen.

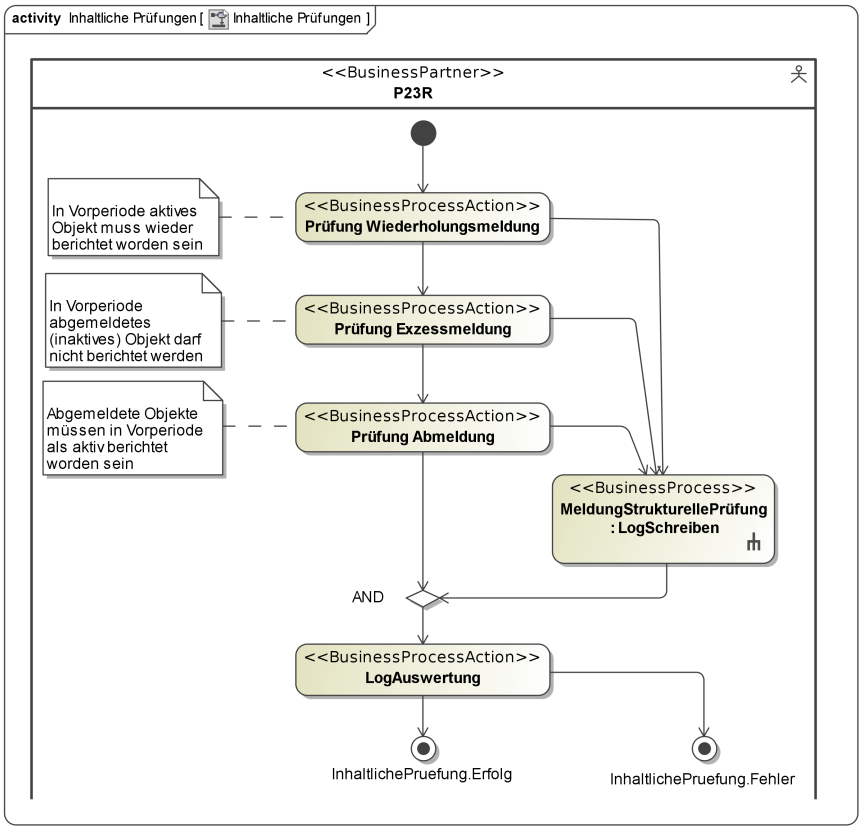


Abbildung C.16.: Aktivitätsdiagramm „Inhaltliche Prüfungen“

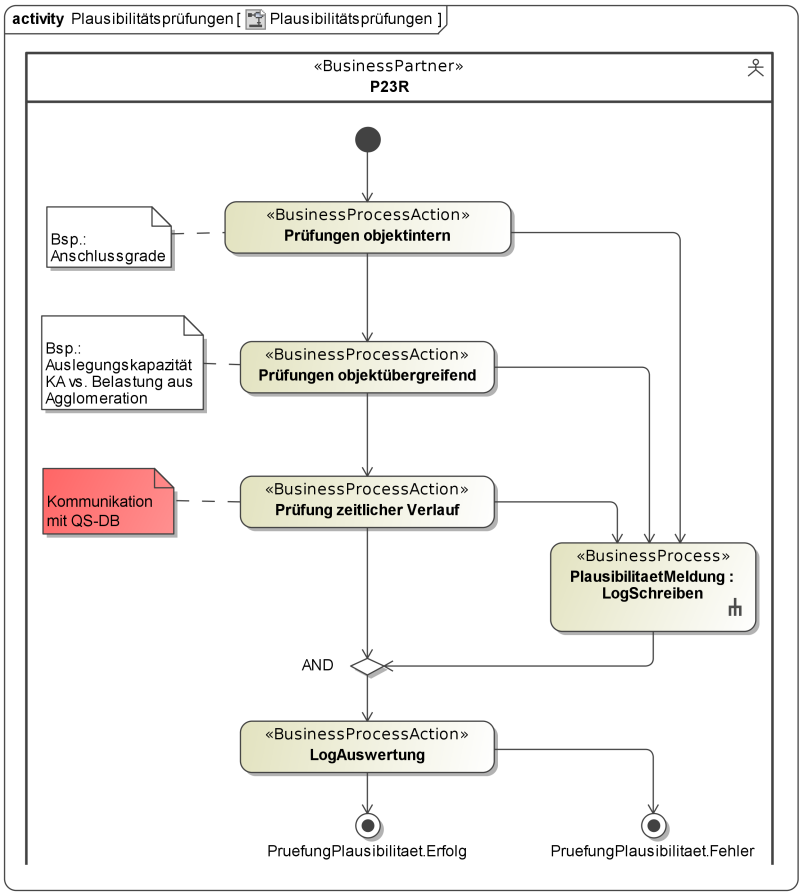


Abbildung C.17.: Aktivitätsdiagramm „Plausibilitätsprüfungen“

C.3. XUKommunalabwasser Tests und Prozesskette

XUKommunalabwasser ist der zertifizierte XÖV-Standard zur Übertragung von Berichtsdaten gemäß EU-Kommunalabwasserrichtlinie 91/271/EWG zwischen Bundesländern und Umweltbundesamt. Erfasst wurden unter XUKommunalabwasser Testfälle, -daten und -nachrichten sowie die Modellierung in einer UML2-Prozesskette mit Prozessmodell, Nachrichtenmodell und P23R-Steckbrief.

C.3.1. Tests XUKommunalabwasser

C.3.1.1. Testfälle

Für die Durchführung von Tests auf einer XUKommunalabwasser P23R-Implementation wurden zunächst Testfälle benötigt. Für die eingangs skizzierte Übertragung von Berichtsdaten waren sinnvolle, weil auf einander abfolgende und den realen Anforderungen entsprechende Testfälle nötig:

- das Versenden einer Anforderung der Kommunalabwasser-Berichte durch das Umweltbundesamt an alle Bundesländer

- das Versenden der Berichtsdaten (Siedlungsgebiete, Kläranlagen, Einleitstellen, Landesinformation) vom Bundesland an das Umweltbundesamt und die Ausführung der damit verbundenen Prüfungen und
- das automatisierte Versenden von Fehler- bzw. Warnmeldungen des Prüfungsmechanismus.

C.3.1.2. Testdaten, Testnachrichten und Testbenachrichtigungen

Nach dem P23R-Prinzip zu übermittelnde Nachrichten bestehen aus einem Teil Metainformation und den zu übertragenden fachlichen Daten, die beim P23R „Payload" genannt werden.
Das P23R-Prinzip verwendet neben dem in der IT üblichen Begriff „Nachricht" für die Übertragung von Daten auch den Begriff „Benachrichtigung". Dabei kommt der P23R Benachrichtigung die Bedeutung zu, tatsächliche Nutzdaten (Antrag, Bericht, Meldung, Statistik) zur Erfüllung einer Pflicht zu transportieren. Eine P23R Nachricht hingegen löst die Generierung einer P23R Benachrichtigung aus oder ist eine Reaktion (Antwort) auf eine P23R Benachrichtigung.

Anforderung der Berichtsdaten durch den Bund
Die Anforderung der Berichtsdaten muss folgende fachliche Informationen (Payload) umfassen:

- Schlüssel, der die aktuelle Berichtsperiode identifiziert
- Datum des ersten Tags des Berichtszeitraums (Anfangsdatum)
- Datum des letzten Tags des Berichtszeitraums (Enddatum)
- Hinweis, ob die EU-Kommission die Berichtsstruktur der aktuellen Berichtskampagne gegenüber der vorherigen Berichtskampagne verändert hat
- Wenn dem so ist: Hinweis (URL) auf die Beschreibung der Änderungen
- Datum des Tages, an dem die Kommunalabwasserdaten spätestens fehlerfrei berichtet sein müssen (Abgabedatum)
- Versionsbezeichner der aktuell gültigen Schnittstellendefinition für die Berichtsdatenlieferung von Bundesland an das Umweltbundesamt

Dabei wurde vorausgesetzt, dass der Berichtskontext (Kommunalabwasser) durch mit übersandte Metainformationen oder die URL des angesprochenen Dienstes bekannt ist. Für den Test einer P23R Regel-Implementierung wurde eine entsprechende Testnachricht erstellt. Ein XML Schema, das die XML-Datei bzgl. ihrer Struktur festlegt, findet sich im Nachrichtenmodell, Abschnitt C.3.2.1.

Übermittlung der Berichtsdaten vom Land an den Bund
Der strukturelle Umfang der Berichtsdaten ist durch das Schema XUKommunalabwassser festgelegt. XUKommunalabwasser umfasst mehrere Dutzend Attribute in folgenden Haupt-Entitäten:

- Kläranlagen-Landesinformation mit Angaben zu Kleinkläranlagen sowie auf Landesebene aggregierten Informationen zum Klärschlamm

- Siedlungsgebiet mit der Beschreibung des Siedlungsabwassers und der Kanalisation
- Kläranlage mit Angaben zur Auslegung, Belastung, Erlaubniswerten und deren Einhaltung und
- Einleitstelle mit Angaben zur Einleitung, zum Einleitgewässer und der Verortung der Einleitstelle bzgl. verschiedener Ordnungssysteme

Für den Test einer P23R Regel-Implementierung wurden zwei XML-Dateien mit Testbenachrichtigungen mit sowohl intakten als auch defekten Testdaten zu jeder Haupt-Entität entwickelt.

Fehler- bzw. Warnmeldungen der QS-Prüfung
Die Struktur der Protokolle (Logs), die die QS-Prüfung erzeugt, musste Auskunft über das Gesamt-Prüfungsergebnis geben. Die für Fehler und Warnungen notwendigen beschreibenden Daten sind durch die UML2-Modellierung der Aktivität „Log (Fehlerprotokoll) Schreiben“ (Abb. C.14) beschrieben. Für den Test einer P23R Regel-Implementierung wurden drei entsprechende Testnachrichten erstellt. Ein XML Schema, das die XML-Dateien mit den Fehler-, Warn- bzw. Erfolgsmeldungen bzgl. ihrer Struktur festlegt, findet sich im Nachrichtenmodell, Abschnitt C.3.2.1.

C.3.2. Prozesskette XUKommunalabwasser

Der logische Ablauf der Prozesskette XUKommunalabwasser ergibt sich aus den Anforderungen der Berichtspflicht. Diese dienten bereits als Grundlage für die Entwicklung der Testfälle.

C.3.2.1. Prozess- und Nachrichtenmodell

Die Übersicht über die Prozesskette XUKommunalabwasser – und damit über das Prozessmodell – ist im Use-Case-Diagramm „Prozesskette XUKommunalabwasser“ (Abb. C.18) dargestellt. Das Aktivitätsdiagramm „XUKommunalabwasser Berichtsanforderung“ (Abb. C.19) bildet die Anforderung der Berichtsdaten zur Kommunalabwasserberichterstattung ab.

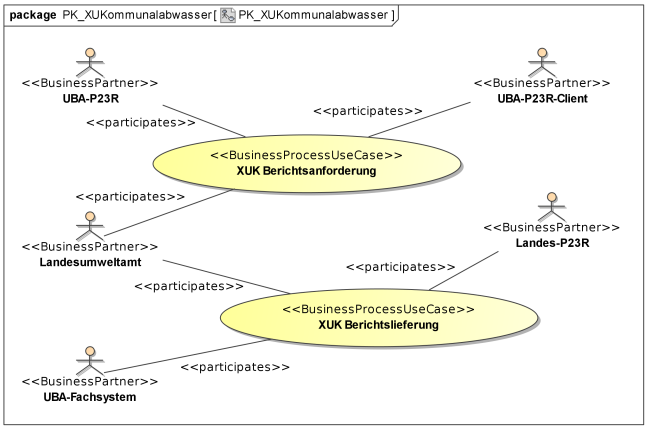


Abbildung C.18.: Use-Case-Diagram „Prozesskette XIKommunalabwasser”

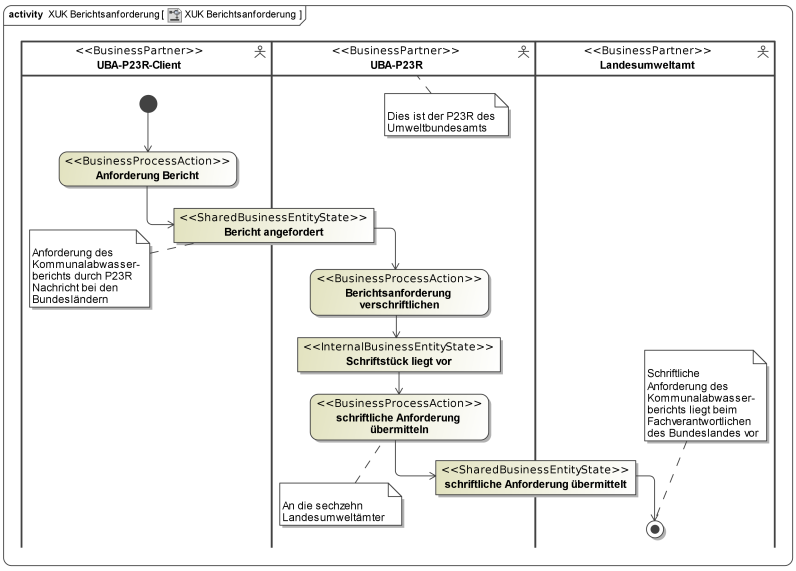


Abbildung C.19.: Aktivitätsgdiagramm „XUKommunalabwasser Berichtsanforderung”

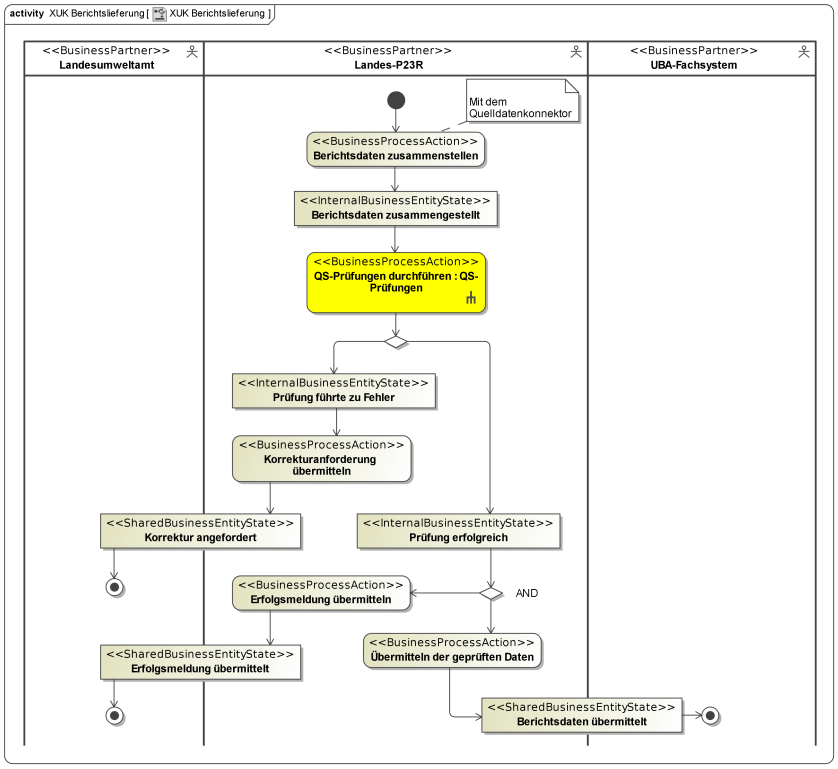


Abbildung C.20.: Aktivitätsgdiagramm „XUKommunalabwasser Berichtlieferung”

Das Nachrichtenmodell besteht aus der Beschreibung der zwischen den Kommunikationspartnern auszutauschenden Nachrichten. Die meisten Nachrichten werden im XML-Format übertragen. Ihr Aufbau wird hier durch XML Schema-Dateien beschrieben.

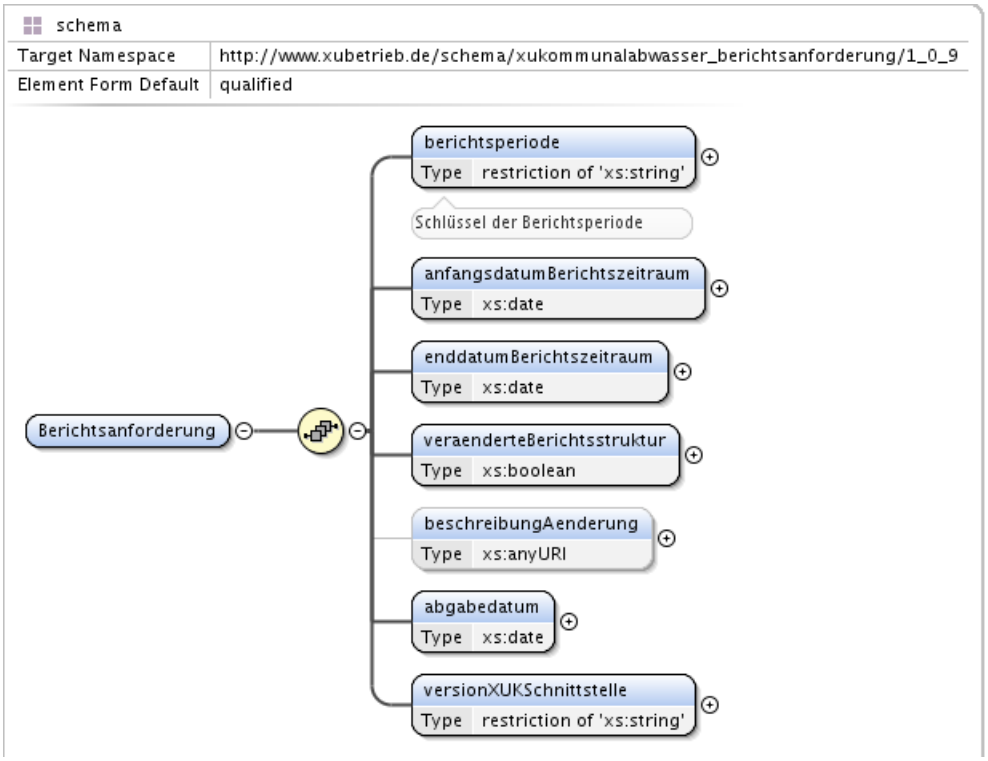


Abbildung C.21.: XML Schema-Darstellung der Anforderung des Berichts

Für das Nachrichtenmodell ist die zwischen dem UBA-P23R-Client und dem UBA-P23R ausgetauschte Nachricht, die mit dem Status „Bericht angefordert“ verbunden ist, von Bedeutung. Sie legt den inhaltlichen Umfang der Berichtsanforderung fest und wird durch ein dafür entwickeltes XML Schema beschrieben. Die grafische Repräsentation des Schemas zeigt Abb. C.21. Der Aufbau der schriftlichen Anforderung, die vom UBA-P23R an das Landesumweltamt übermittelt wird, ist durch das Umweltbundesamt festzulegen. Dabei müssen zwangsläufig auch die in der XML-Nachricht übertragenen Daten aufgeführt werden, damit das Landesumweltamt seiner Berichtspflicht fristgerecht nachkommen kann. Da im P23R die Erstellung von PDF-Dokumenten vorgesehen ist, sollte diese Funktionalität hier eingesetzt werden, um die Anforderung zu verschriftlichen. Wenn das P23R-Prinzip auf Seite der Bundesländer eingesetzt wird, sollte die Berichtsanforderung auch als P23R-Nachricht in das Vorgangsverwaltungssystem des jeweiligen Landesumweltamts übermittelt werden, sofern ein solches vorhanden ist. Dabei käme das selbe oben genannte Schema zum Einsatz. Es kann dann theoretisch auch auf die verschriftlichte Berichtsanforderung verzichtet werden, die auch vorher nicht ausgedruckt werden muss. Das Aktivitätsdiagramm „XUKommunalabwasser Berichtslieferung“ (Abb. C.20) bildet den Liefer- und Prüfprozess des Berichts ab. Für das Nachrichtenmodell sind sowohl die zwischen dem Landes-P23R und dem Landesumweltamt ausgetauschten Nachrichten über Korrekturnotwendigkeiten und die Mitteilung des Erfolgs von Bedeutung als auch der vom Landes-P23R im Erfolgsfall an das UBA-Fachsystem übermittelte Bericht. Sowohl für die Korrekturnotwendigkeit als auch die Erfolgsmeldung wird das dafür entwickelte XML Schema verwendet. Die grafische Repräsentation des Schemas zeigt Abb.C.22.

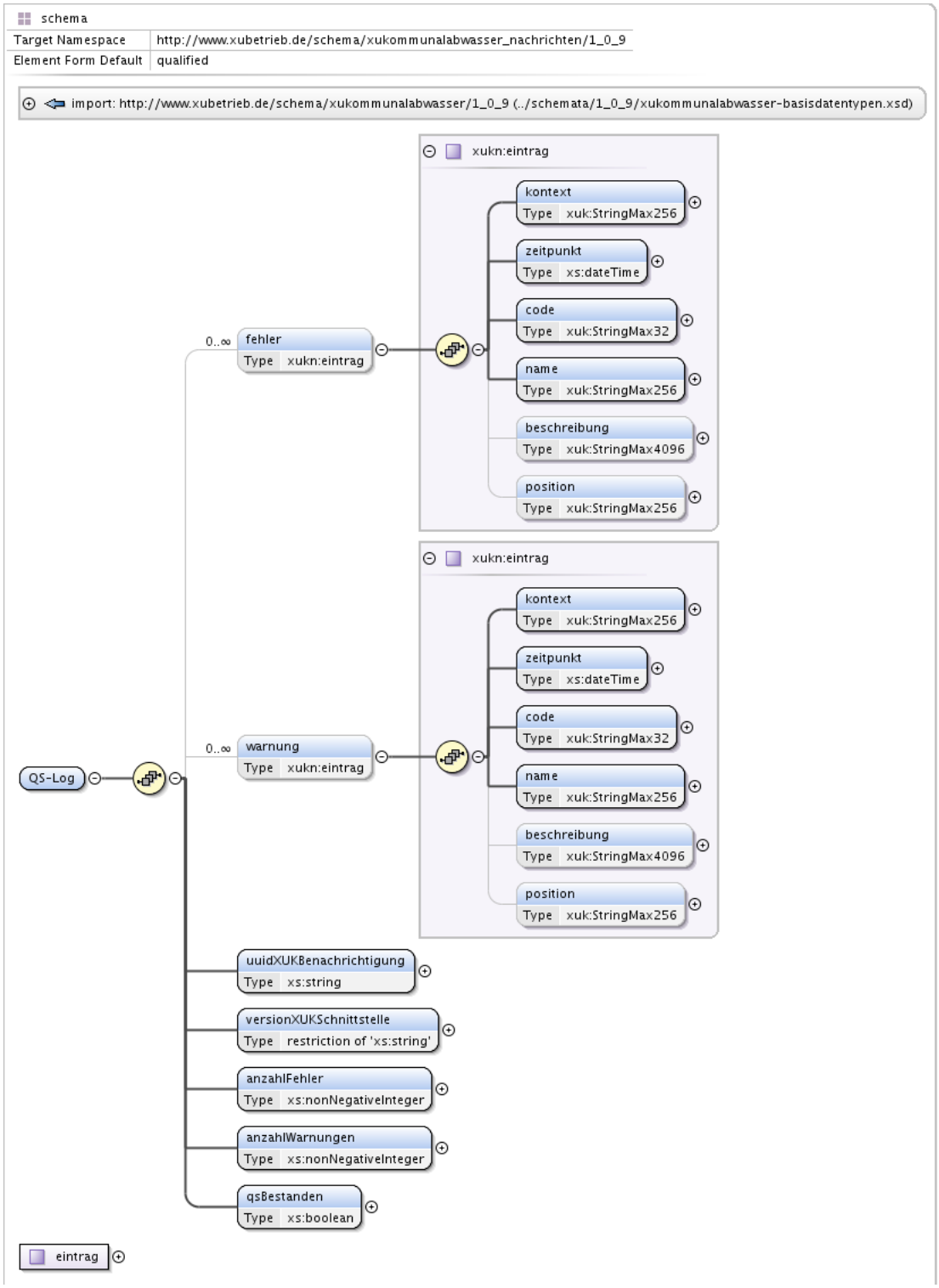


Abbildung C.22.: XML Schema-Darstellung der Prüfmeldung des Berichts

Für die Übermittlung des Berichts wurde das XML Schema aus dem XÖV-Standard XUKommunalabwasser verwendet. Dieses Schema eignete sich aufgrund seines Umfangs und der Tatsache, dass es automatisiert aus UML2-Modellen abgeleitet wird,

nicht für die grafische Darstellung[2].

C.3.2.2. Prozessketten-Steckbrief

Das Glossar des P23R Methodenleitfadens beschreibt den Prozessketten-Steckbrief (PRKSteckbrief) wie folgt: „In den auf der Methodik des Methodenleitfadens basierenden PRK-Steckbriefen werden alle grundlegenden Merkmale (Gesetzesgrundlage, vorhandene Fachverfahren, etc.) einer bestimmten Meldepflicht erfasst." Der Aufbau und die Interpretation der Attribute des Prozessketten-Steckbriefs wird in Modul 1[3] des Methodenleitfadens beschrieben.

PRK-Attribut	**Wert**
Laufnr. (M1)	-
Name der PRK / Titel der Informations- und Meldepflicht (M2)	Kommunalabwasserberichterstattung gemäß EU-Richtlinie 91/271/EWG
Kontakttyp (M3)	Bericht
Wertschöpfungscluster (M4)	betriebliche Umweltberichterstattung
Beteiligte Akteure auf Wirtschaftsseite (M5)	An die Stelle der Wirtschaftsseite treten hier die Landesumweltämter
Beteiligte Akteure auf Behördenseite (M6)	Umweltbundesamt
Zusätzliche Beteiligte (M7)	-
Bereich des B2G Kontaktes (M8)	Richtlinie 91/271/EWG
Typische Informationselemente (M9)	Siedlungsabwasser-, Kläranlagen- und Einleitstellendaten; landesweit aggregierte Informationen zu Nährstofffrachten, Kleinkläranlagen und Schlammverwendung
Inhaltstypen (M10)	Ganzzahlen, Fließkommazahlen, bereichsbeschränkte numerische Werte (z. B. Prozentangaben), Listen-(Katalog-)werte, Texte verschiedener Maximallängen, Datumsangaben, boolesche Werte, bezüglich Mustern beschränkte Zeichenketten (z. B. Versionsangaben)
Prozessmodellierung (M11)	UML2 mit MagicDraw Version 16.5
Gesetzliche Grundlage (M12)	EU-Richtlinie 91/271/EWG
Gesetzliche Änderungen (M13)	Es sind keine geplanten Änderungen bekannt
IP Identnr. in der Standardkostenmodell (SKM) Datenbank (M14)	-
Fallzahlen nach SKM (M15)	-
Periodizität oder Frequenz [a-1] (M16)	0,5/a
Bürokratiekosten nach SKM (M17)	-

[2] Eine interaktive, grafische Darstellung der Version 1.0.0 des Schemas findet sich hier: http://xml.xubetrieb.de/xukommunalabwasser/ver1.0.0/xukommunalabwasser-nachrichten.html#id72

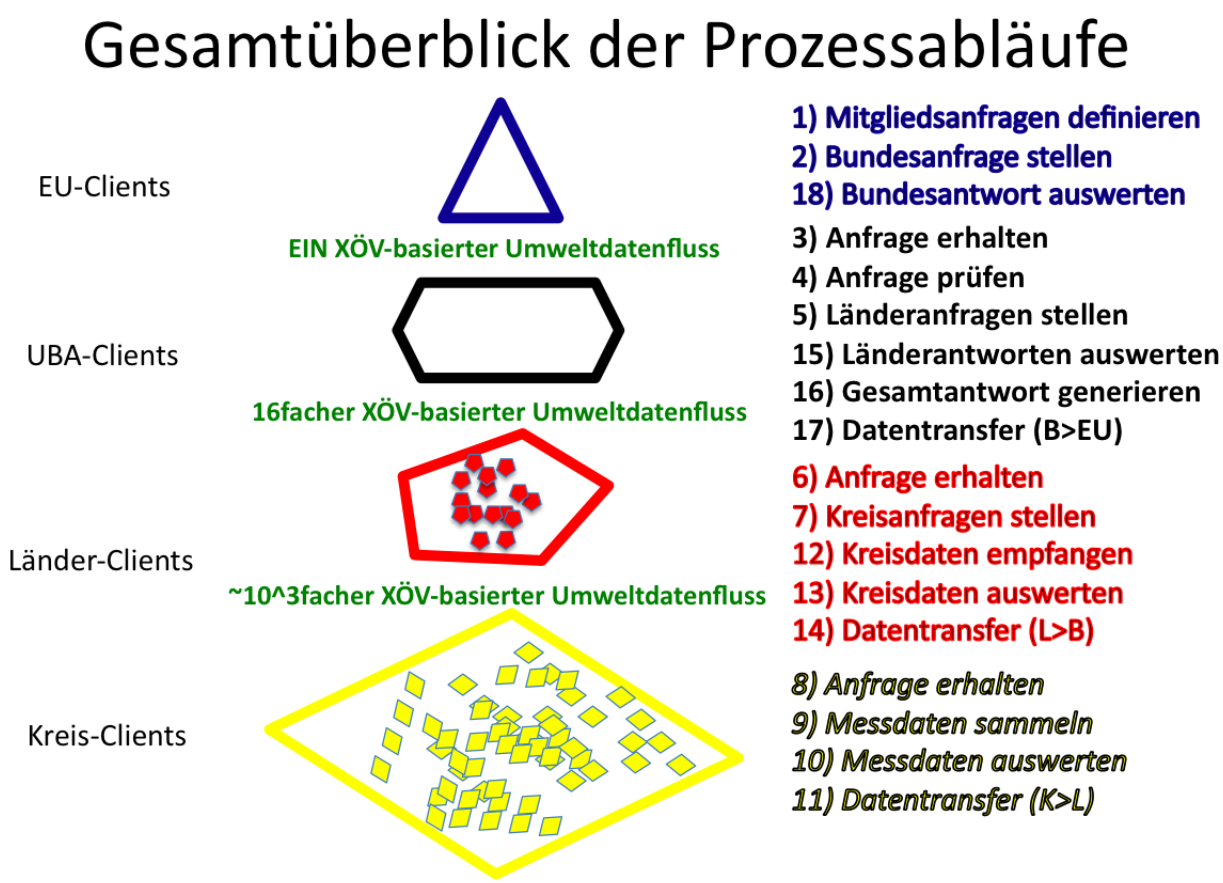
[3] Stand 18.03.2013

Informationssystem auf Unternehmensseite (M18)	An die Stelle der Unternehmensseite treten hier die Landesumweltämter. LfU Bayern: DABay (Datenverband Abwasser Bayern); Sachsen: Datenbank kommunales Abwasser, FIS-WRV (Fachinformationssystem wasserrechtlicher Vollzug) und ADÜ (Datenbank Abwasserdirekteinleiterüberwachung)
Informationssystem auf Behördenseite (M19)	UDIS e-Kommunalabwasser
Informationssystem – Schnittstellen – Rückkanalfähigkeit (M20)	UDIS e-Kommunalabwasser verfügt über ein bidirektionales RESTful XUKommunalabwasser- Interface, welches für die Payload der P23R-Benachrichtigung eingesetzt werden kann
Medium (M21)	Derzeit noch Webplattorm mit Unterstützung von Uploads, Online-Eingabe und Prüfungen
Datenformat (M22)	XML und CSV; XML: XÖV-zertifizierter Standard XUKommunalabwasser Version 1.0.9 auf Basis des XÖV-Standards XUBetrieb, CSV: e-Kommunalabwasser Wiki
Transportprotokoll (M23)	htpps
Digitales Datenmodell (M24)	XUKommunalabwasser Version 1.1.6

C.4. Testarchitektur für prozessorbasierte Nachrichtengruppen im Umweltsektor

Informationstechnologische Testmuster zur Evaluierung von P23R-basierten Nachrichtengruppen

Für die Rahmenbedingungen des P23R-Projekts wurde im Folgenden ein informationstechnologisches Modell entwickelt, auf dessen Basis generische Testmuster für reale Testfälle/Prozesse P23R-basierter Nachrichtengruppen abgeleitet werden konnten. Dabei lag der Fokus auf der Client-P23R-Kommunikation mit Hilfe von zuvor definierten XMLNachrichtengruppen. Der untersuchte Gesamtgeschäftsvorfall auf Seiten des Umweltbundesamtes hatte das primäre Ziel, der Europäischen Union qualitativ hochwertige Daten zeitnah zur Verfügung zu stellen. Dabei mussten vielfältige Prozessabläufe auf Bundes-, Länder- und kommunalerbzw. Kreisebene berücksichtigt werden:



Die einzelnen Geschäftsvorfälle zur automatisierten Berichtserstellung unter Einbezug aller Ebenen sind in der vorherigen Abbildung idealisiert dargestellt. Die konkreten Prozessdatenflüsse wurden erfasst und mittels UML- und Datenmodellen maschinenlesbar dokumentiert. Das volle Potenzial der P23R-Datentransferidee könnte auf allen vier Verwaltungsebenen entfaltet werden – eine verschachtelte Nutzung über mehrere legislative Ebenen ist möglich und sinnvoll. Mittels der entsprechenden Clients und deren Adaptern (mit den zu implementierenden Quell- und Zielkonnektoren) kann ein Nachrichtenaustausch mit dem P23R-Prozessor nach folgenden generischen und wiederkehrenden Schritten erfolgen:

1. Client sendet eine Initialisierungsnachricht an den P23R-Prozessor
2. P23R-Prozessor sendet die finalisierten Dokumentendaten als Überprüfungsnachricht an den Client
3. Nach Bestätigung durch den Fachanwender sendet der Client eine „ok“-Nachricht an den P23R-Prozessor, um den authentifizierten Datentransfer XÖV-basierter Umweltdaten zu starten.

Zur Beurteilung der Vollständigkeit der Kommunikation nach dem P23R-Prinzip wurden nach Rücksprache mit dem Auftraggeber informationstechnologische Muster entwickelt, anhand derer eine zukünftige Kommunikation nach festen Kriterien bzgl.

ihrer Vollständigkeit beurteilt werden kann. Die Beurteilungskriterien und ihre zugrunde liegenden ITRegeln spiegeln somit auch die grundlegenden Funktionsprinzipien einer jeden zentral orientierten Kommunikationsarchitektur wieder.

C.4.1. IT-Regelklassen

Unter einer informationstechnologischen Regel (IT-Regel) wird die exakt wiederholbare automatisierte Generierung eines Bitmusters verstanden. Genügt die Implementierung einer IT-Regel den jeweiligen Systemanforderungen, erzeugt diese IT-Regel auf dem jeweiligen System immer das gleiche Ergebnis auf Prozessorebene. IT-Regeln lassen sich gemäß ihres informationslogischen Zweckes in unterschiedliche Klassen oder Gruppierungen einteilen. Aus der Beschreibung des minimalen Sets an Regelgruppen und des informationstechnologischen Zusammenspiels dieser lassen sich für konkrete QS-Prozesse im Umweltsektor logische Testmuster ableiten, denen alle sinnvollen P23R-Implementierungen genügen müssen.
Dadurch wurde es möglich, konkret testen zu können, ob eine mögliche P23RImplementierung den informationstechnologischen Qualitätsansprüchen der QSProzesse im Umweltsektor genügt. Folgende IT-Regelgruppen werden mindestens benötigt, um die evaluierten QS-Prozessbausteine im Umweltsektor informationstechnologisch abbilden zu können:

1. Kontextregeln (definieren den administrativen Kontext)
2. Prozessregeln (Verknüpfung der Bausteine)
3. Datenstrukturregeln (jeweiliger Inhalt der Nachrichten)
4. Nachrichtenregeln des P23R-Client-Systems
5. Antwortgenerierungsregeln des P23R-Master-Systems
6. Regeln zum Kommunikationsmedium (Transportprotokolle wie OSCI, TCP/IP, etc.)
7. Mappingregeln (z.B. Datenbankkonnektoren, Serviceadapter, XSLT-Regeln)
8. Systeminitialisierungsregeln für Client und Master
9. Regeln zur Etablierung einer synchronen Zeitmetrik

Werden die oben aufgeführten IT-Regelgruppen auf das Beispiel der Richtlinie 91/271/EWG angewandt, so erhält man das folgende informationstechnologische Testmuster:

1. „Anforderungen Bund“ bzgl. 91/271/EWG
2. Start(Datenanforderung) -> Import -> Prüfung -> Export -> Finalisierung
3. receiving_area, discharge_point, agglomertion, agglomeration_plant, plant, federal_ state_info
4. Umweltbundesamt_client, Landesumweltamt_client
5. activity AnforderungBund

6. Die vom Client zu verwendenden Transportprotokolle
7. Die vom Client zu verwendenden Mappingregeln und Konnektoren
8. –
9. Zeitstempel-Tag

Um zu Testaussagen über die zukünftig dem P23R-Prinzip folgende Kommunikation gelangen zu können, war eine Überführung der IT-Regeln in eine maschinenlesbare Struktur notwendig. Der hier gewählte generische Ansatz generiert eine logisch notwendige XML-Struktur, die für alle möglichen Nachrichten stabil ist. Aus Perspektive der QS-Prozesse im Umweltsektor sind die infrastrukturtechnischen IT-Regeln 5, 6, 7, 8 und 9 von sekundärer Bedeutung und wurden folglich nicht weiter in der Modellierung berücksichtigt.

Typ 0: Generische XML-Nachricht

```
<?xml version="1.0" encoding="UTF-8"?>
<xbn:xmlBasedNotification version="2013.feb"
   xmlns:xbn= "notificationSchemaURI"
    xmlns:xbd= "dataSchemaURI">
<xbn:modelDesigner author="BINFORT␣GmbH"/>
<xbn:genericMessagePattern>
    <xbd:userID/>
    <xbd:clientSenderID/>
    <xbd:clientReceiverID/>
    <xbd:messageID/>
    <xbd:transferDATA/>
</xbn:genericMessagePattern >
</xbn:xmlBasedNotification >
```

Die exemplarische Anwendung dieser generischen XML-Struktur für den Geschäftsprozess 91/271/EWG lieferte umweltprofilierte XML-Testmusterstrukturen für die drei benötigten Kommunikationsprozesse zwischen Client und Master: P23R-Initialisierung, P23R-Antwort und P23R-okNachricht:

Typ 1: Initialisierungsnachricht:

```
<?xml version="1.0" encoding="UTF-8"?>
<xuba:UBAnotification version="2014.feb"
xmlns:xuba="http://www.UBA.de/xoevNotifications"
xmlns:xukommunalabwasser="http://www.xubetrieb.de/schema/xukommunalabwasser/1_1_2">
<xuba:p23rInitialisierung_91/271/EWG>
    <xuba:userID profile="UBA␣Fachgebiet␣II␣2.2"/>
    <xuba:clientSenderID profile="uba822"/>
    <xuba:clientReceiverID profile="eu38"/>
    <xuba:messageID profile="AnforderungenBund2014#1.1"/>
    <xuba:transferDATA profile="xukommunalabwasser:AbwasserRezyklierung"/ >
</xuba:p23rInitialisierung_91/271/EWG>
</xuba:UBAnotification>
```

Typ 2: Antwortsnachricht:

```
<?xml version="1.0" encoding="UTF-8"?>
<xuba:UBAnotification version="2014.feb"
    xmlns:xuba="http://www.UBA.de/xoevNotifications"
xmlns:xukommunalabwasser="http://www.xubetrieb.de/schema/xukommunalabwasser/1_1_2">
<xuba:p23rAntwort_91/271/EWG>
    <xuba:clientReceiverID profile="uba822"/>
    <xuba:messageID profile="AnforderungenBund2014#1.2"/>
    <xuba:transferDATA profile="xmlns:xukommunalabwasser"/>
  </xuba:p23rAntwort_91/271/EWG>
</xuba:UBAnotification>
```

Typ 3: ok-Nachricht:

```
<?xml version="1.0" encoding="UTF-8"?>
<xuba:UBAnotification version="2014.feb"
    xmlns:xuba="http://www.UBA.de/xoevNotifications">
    <xuba:p23rOKnachricht_91/271/EWG>
      <xuba:userID profile="UBA␣Fachgebiet␣II␣2.2"/>
      <xuba:clientSenderID profile="uba822"/>
      <xuba:messageID profile="AnforderungenBund2014#1.3"/>
      <xuba:transferDATA profile="ok"/>
     </xuba:p23rOKnachricht_91/271/EWG>
</xuba:UBAnotification>
```

C.5. P23R-konforme Umsetzung von XUKommunalabwasser

Um zu einer P23R-konformen Umsetzung der Kommunalabwasserberichterstattung (XUKommunalabwasser) zu kommen, werden neben der Entwicklung der jeweiligen P23R-Regel folgende Aufgaben durchgeführt:

- Erläuterung der Berichtsszenarien und Herstellen des notwendigen Projektverständnisses bei den Kooperationspartnern
- Unterstützung der Kooperationspartner mit spezifisch auf die einzelnen QSPrüfungen zugeschnittenen Testdaten und zugehörigen Testergebnissen
- Erstellen der P23R SourceConnectoren, die die zu berichtenden Daten im XML-Format bereit stellen
- Erstellen der P23R CommunicationConnectoren, die die von der jeweiligen P23RRegel bereit gestellten Daten an den Datenempfänger übermitteln
- Herbeiführen der Lauffähigkeit eines P23R-Prozessors, der Source- und CommunicationConnectoren im Realeinsatz nutzt

C.5.1. Szenarien und Testdaten

Das im Vorgängervorhaben vorbereitete Szenario für die Kommunalabwasserberichterstattung (Kapitel C.3) wird um spezielle Testdateien für die abwasserfachlichen Qualitätsprüfungen erweitert.

Um dem Berichtslieferanten eine strukturierte Vorgehensweise bei der Verfolgung von Berichtsmängeln zu bieten, werden die möglichen fachlichen Berichtsfehler bzw. -warnungen in gleicher Weise gegliedert wie die Prüfungen. Dazu wird jeder Fehlersituation ein aussagekräftiger Name und eine Beschreibung zugeordnet. Die Position des Fehlers wird durch die Benennung des Objekts mit der fehlerhaften Information übermittelt. Dieses kann der Bearbeiter i.d.R. (dies ist von der jeweils verwendeten IT-Lösung abhängig) sofort lokalisieren. Eine Zeilennummer ist bei automatisiert erstellten Dateien – was bei der Übermittlung von XML-Daten der Regelfall ist – nicht hilfreich für die Lokalisierung eines fachinhaltlichen Fehlers.

C.5.2. Connectoren

Die Entwicklung sowohl der SourceConnectoren als auch der CommunicationConnectoren wird hier unter Einsatz der Skriptsprache Perl durchgeführt. Aufgabe der SourceConnectoren (SC) ist die Bereitstellung von Eingangsdaten für die P23R-Regel.
Für die Berichterstattung XUKommunalabwasser werden zwei SC entwickelt. Der erste (SC1) liefert die zu berichtenden Daten aus der Datenhaltung des liefernden Bundeslandes. Der zweite (SC2) liefert Berichtsdaten der Vorperiode, die für die Prüfung der Objekthistorie benötigt werden.
Im Übertragungsprozessen XUKommunalabwasser ist vorgesehen, dass der Datenlieferant in jedem Fall und der Datenempfänger in keinem Fall das Prüfprotokoll erhält. Im Fall einer Prüfung ohne erkennbare Qualitätsmängel der Daten erhält der Berichtsempfänger die für ihn bestimmten Daten. Treten bei der Prüfung Qualitätsmängel an den Daten zutage, erhält der Datenlieferant die Daten zusammen mit dem Prüfprotokoll zurück. Gemäß dem P23R-Prinzip werden diese Aufgaben je Berichtspflicht jeweils von zwei CommunicationConnectoren (CC) durchgeführt, die sich in ihrem internen Aufbau jedoch nur wenig unterscheiden. Um den wirkbetriebsfähigen P23R Berichtsprozess für XUKommunalabwasser herzustellen, wird eine Demonstrationsumgebung (Kap. G.5) aufgebaut.
Für XUKommunalabwasser greift SC1 auf die Daten des Landes (Fachsystem Land, Fasy-Land) zu und erstellt auf Anforderung durch die P23R-Regel die aktuell zu berichtenden XML-Daten. SC2 greift auf die vom Umweltbundesamt gespeicherten Berichtsdaten zu und erstellt die XML-Daten der vorherigen Berichtsperiode. Diese werden in einem gemeinsamen Dokument an die P23R-Regel gereicht (Regeleingang), die in der Folge die notwendigen QS-Prüfungen durchführt und ein XML-Dokument mit dem Prüfprotokoll und den Berichtsdaten erstellt (Regelausgang). Nach der Freigabe der Daten vom Regelausgang durch den Datenlieferanten wird abhängig vom Ausgang der QS-Prüfung nun der CC1 (Erfolg) oder der CC2 (Qualitätsmängel festgestellt) mit der weiteren Verarbeitung der Daten von der P23R-Regel beauftragt. Die CC erstellen in diesem Projekt in den meisten Fällen E-Mails, deren Anhang aus der auszuliefernden XML-Datei besteht. Ausnahme eins (Abb. C.23) ist der Fall der Anforderung von Berichtsdaten zu XUKommunalabwasser. Hier wird der XML-Inhalt in eine menschenlesbare E-Mail umgewandelt und an Bundeslandempfänger versandt.

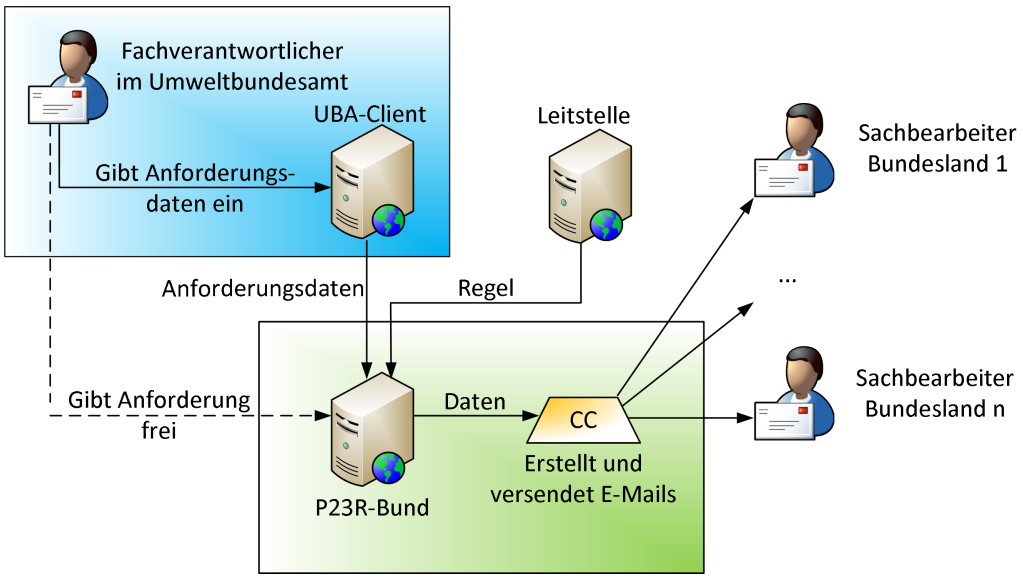


Abbildung C.23.: Datenanforderung durch UBA bei den Bundesländern

Ausnahme zwei (Abb. C.24) ist die XUKommunalabwasser Berichtsmeldung (Erfolgsfall), bei der die Berichtsdaten vom CC an einen Importdienst (RESTful Importservice, produktiv) geleitet werden, der auf der zentralen Datenbank des Abwasserkatasters UDIS des Umweltbundesamts einen Import dieser Daten durchführt.

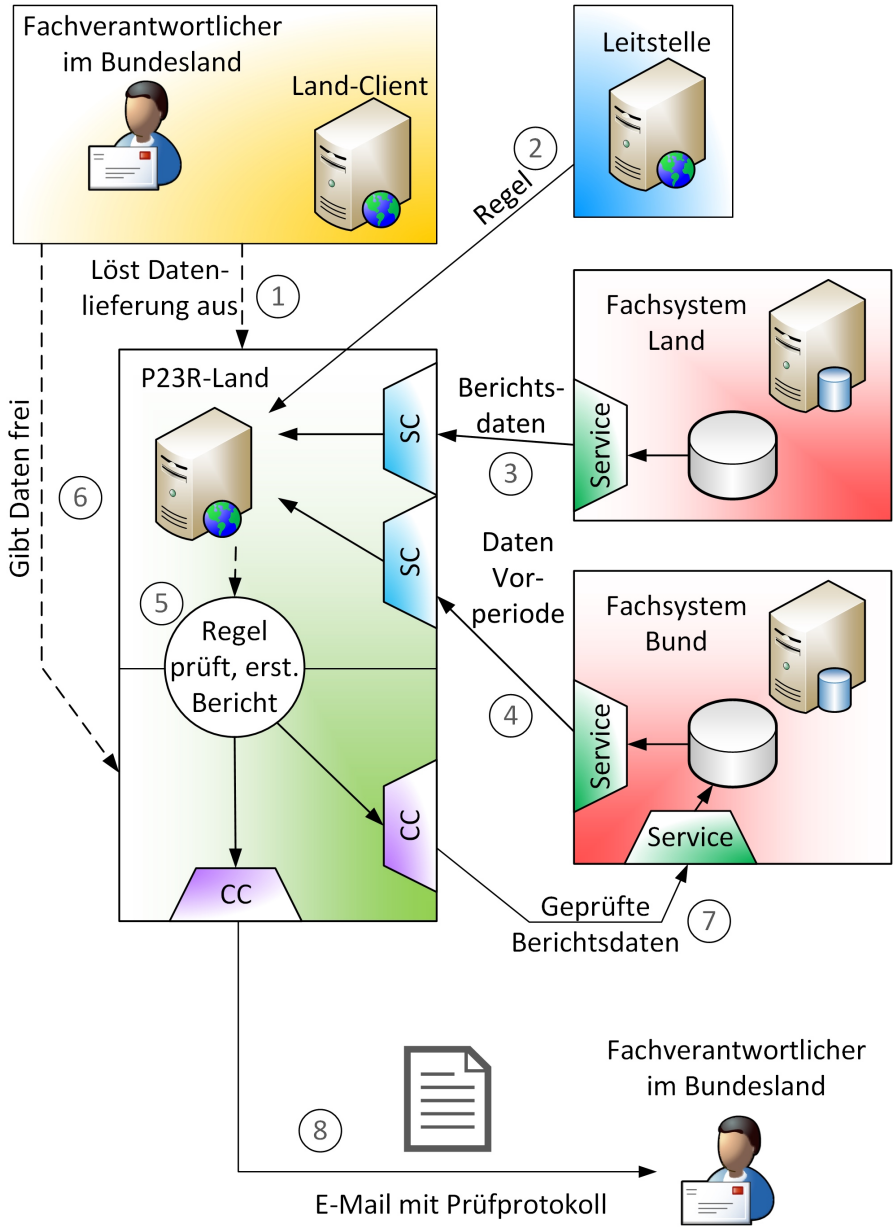


Abbildung C.24.: Datenanforderung durch UBA bei den Bundesländern

D. Prozess §61 WHG

Die folgende Kapitel D ist dem Abschlussbericht der ENDA KG entnommen. Die Regel steht ab Juli 2014 zum Download unter http://p23r.enda.eu bereit.

D.1. Prozessmodellierung

D.1.1. Prozessmodellierung für das Bundesland Bayern

Die für die Betrachtung des §61 WHG wesentlichen Anwendungsfälle sind im Use-Case-Diagramm „Kläranlagenjahresbericht“ (Abb. D.1) und den zugehörigen Aktivitätsdiagrammen „Kläranlagenjahresbericht anfordern“ (Abb. D.2) und „Kläranlagenjahresbericht erstellen“ (Abb. D.3) beschrieben. Der Jahresbericht ist jeweils zum 1. März des Folgejahres dem Wasserwirtschaftsamt vorzulegen. Nur wenn dies nicht erfolgt, wird die Aktivität „Kläranlagenjahresbericht anfordern“ ausgelöst. Dabei wird die kommunale Verwaltungsbehörde (KVB) eingebunden, da nur diese die rechtliche Grundlage besitzt, den Jahresbericht einzufordern. Grundlage für den Jahresbericht sind die in den Betriebstagebüchern erfassten Informationen zur Anlage. Bei IT-gestützten Betriebstagebüchern werden teilweise automatisierte Prüfungen vorgenommen, die zwischen den jeweiligen, meist kläranlagenindividuellen Anwendungen, unterschiedlich ausfallen. Es werden Berechnungen durchgeführt, die die Plausibilität von numerischen Werten durch Vergleiche von numerischen Werten miteinander bewerten und die auf starke Abweichungen numerischer Werte über die Zeit prüfen. Ein zentrales Verzeichnis dokumentierter Prüfungen liegt derzeit nicht vor. Im Einzelfall werden auch Korrekturen der Werte durchgeführt. Das vom Betreiber geführte Betriebstagebuch mit dem technischen Anlagenzustand und den detaillierten Überwachungsergebnissen wird nicht übermittelt. Einzelüberwachungsergebnisse, z. B. aus kontinuierlichen und diskontinuierlichen Messungen werden zu Monats- und Jahreswerten zusammengefasst. Rechtliche Grundlage zur Erfassung der Daten (Konzentrationen, Abwassermengen etc.) ist der Genehmigungsbescheid. Der Jahresbericht ist bezüglich seines strukturellen Aufbaus i.S. einer exakten Beschreibung des Inhalts bisher nicht spezifiziert. An einer solchen Spezifikation, ggfs. mit Empfehlungscharakter, besteht Interesse.
Dem Aktivitätsdiagramm „Kläranlagenjahresbericht erstellen“ (Abb. D.3) ist zu entnehmen, dass der Jahresbericht von einem privaten Sachverständigen der Wasserwirtschaft (PSW) geprüft wird. Art und Umfang dieser Prüfung sind nicht geregelt. Es handelt sich um eine Prüfung nach bestem Willen und Gewissen. Ist der Jahresbericht zu beanstanden, wird bzw. werden die Folgeprüfung(en) von der unteren Wasserbehörde durchgeführt, um den pauschal entlohnten PSW nicht über Gebühr zu belasten.

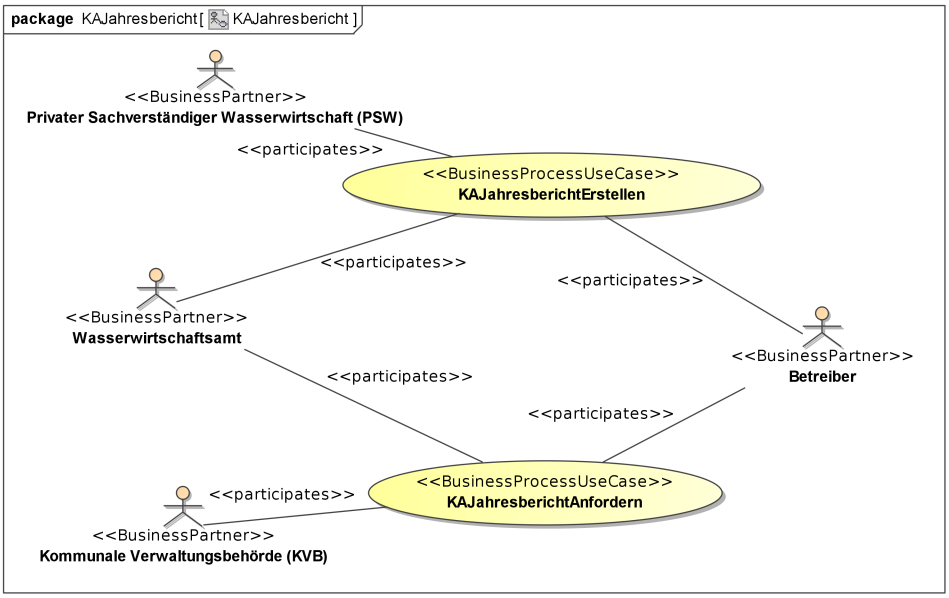


Abbildung D.1.: Use-Case-Diagramm: Kläranlagenjahresbericht”

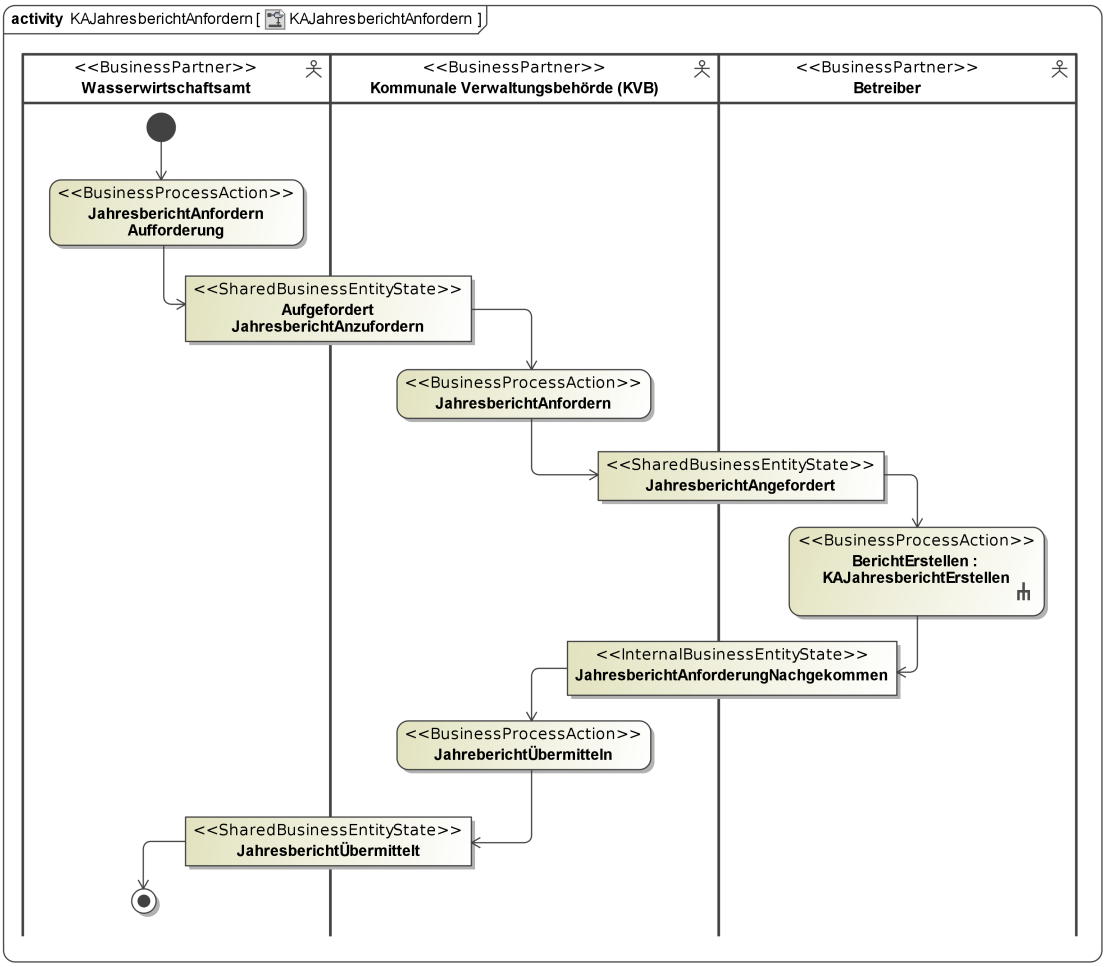


Abbildung D.2.: Aktivitätsdiagramm: „Kläranlagenjahresbericht anfordern”

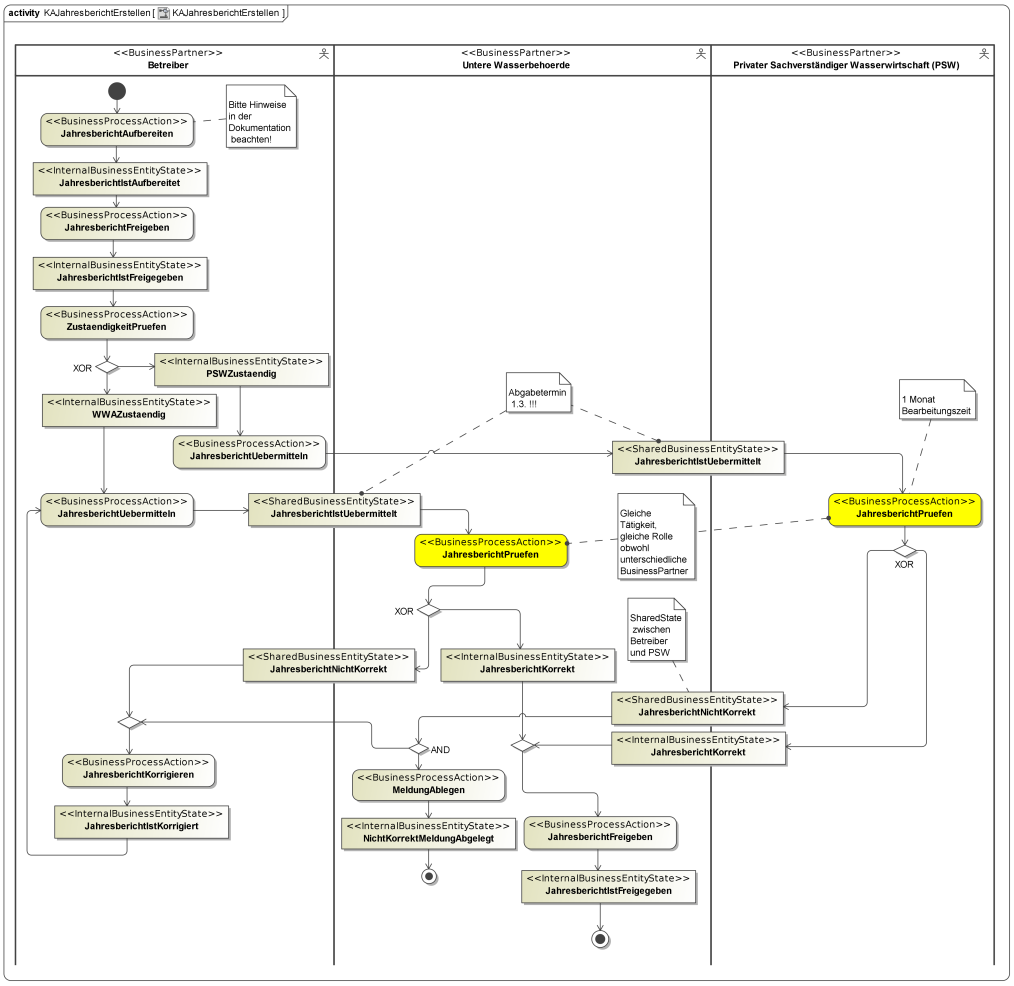


Abbildung D.3.: Aktivitätsdiagramm: „Kläranlagenjahresbericht erstellen”

D.1.2. Prozessmodellierung für das Bundesland Sachsen

Wie auch in Bayern, ist in Sachsen die Erstellung eines Kläranlagenjahresberichts für Abwasseranlagen vorgeschrieben. Rechtliche Grundlage des Jahresberichts ist die Sächsische Eigenkontrollverordnung (die Landesverordnung zur Eigenkontrolle von Abwasseranlagen), die die Erstellung des Jahresberichts für Kläranlagen vorschreibt. Für Kläranlagen ab 5000EW muss die Übermittlung des Jahresberichts an die zuständige Behörde jeweils bis zum 31. März des Folgejahres erfolgen. Bei Kläranlagen mit geringerer Belastung, also zwischen 2000EW und 4999EW, müssen die Jahresberichte jeweils beim Betreiber angefordert werden.
Die sächsische Eigenkontrollverordnung umfasst neben kommunalen Kläranlagen auch industrielle Kläranlagen und Kanalnetze. Sowohl Industriekläranlagen als auch Kanalnetze sind im Hinblick auf die Berichtspflichten gegenüber der Europäischen Kommission bzw. Abbildung D.4 dem Bund derzeit von geringer Bedeutung und werden daher hier nicht näher betrachtet. Der Jahresbericht ist bezüglich seines strukturellen Aufbaus i.S. einer exakten Beschreibung des Inhalts bisher nicht spezifiziert. An einer solchen Spezifikation, ggfs. mit Empfehlungscharakter, besteht In-

teresse. Allerdings sind inhaltliche Struktur und der Umfang der Jahresberichte vom sächsischen Umweltministerium im Entwurf normiert. Dieser Entwurf muss jedoch mit dem Städte und Gemeindetag abgestimmt werden, um breite Akzeptanz zu finden. Der Fachverantwortliche des sächsischen Landesamts verwies bzgl. der Struktur der Jahresberichte auf die von einem in der Wasserwirtschaft einschlägigen Verlag angebotenen Betriebstagebuchvorlagen. Eine solche wurde beschafft und ausgewertet. Die Ergebnisse sind in die Aktivität „Selbstüberwachung gemäß §61 WHG“ (Abb. 3D.4) eingeflossen.

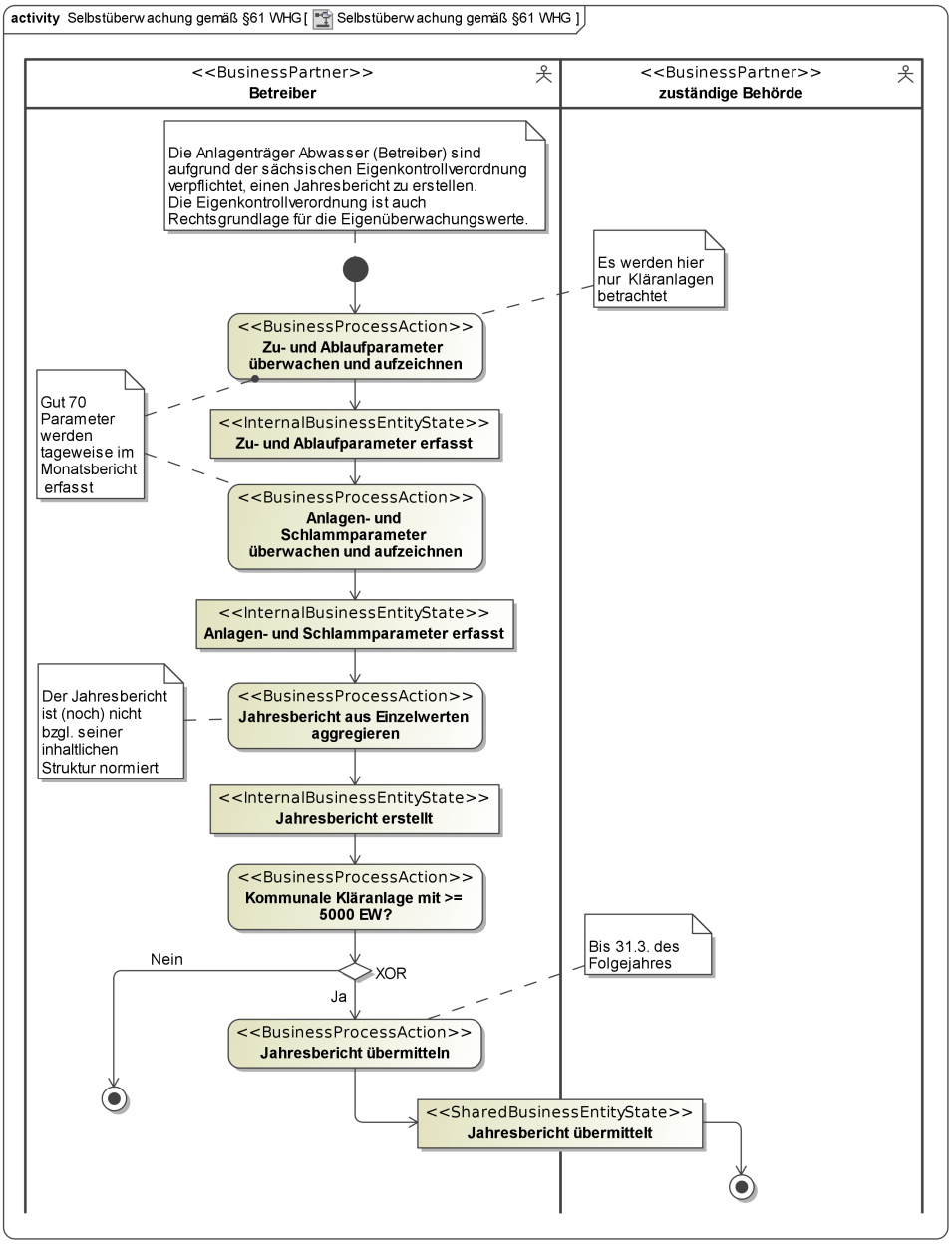


Abbildung D.4.: Aktivitätsdiagramm „Selbstüberwachung gemäß §61 WHG“ Sachsen

D.1.3. Datenmodell

Das Datenmodell xuka_selbstueberwachung lag im Herbst 2013 in der aktuellen Version 1.0.1. vor. Alle Downloads befinden sich unter folgenden Internetadressen:

- Datenmodell: `http://xml.xubetrieb.de/xuka_selbstueberwachung/p61_bundle_rev3.zip`
- Grafische Repräsentation der Benachrichtigung: `http://xml.xubetrieb.de/xuka_selbstueberwachung/ver1.0.1/xuka_selbstueberwachung_nachrichten.html`
- Prüfprotokoll: `http://xml.xubetrieb.de/xuka_selbstueberwachung_log/ver1.0.1/xuka_selbstueberwachung_log.html`
- einzelne Schemata: `http://www.xubetrieb.de/schema/xuka_selbstueberwachung/1_0_1http://www.xubetrieb.de/schema/xuka_selbstueberwachung_log/1_0_1`

D.2. P23R-konforme Umsetzung von §61 WHG

Um zu einer P23R-konformen Umsetzung der Berichterstattung nach §61 Wasserhaushaltsgesetz (WHG) zu kommen, werden neben der Entwicklung der jeweiligen P23R-Regel folgende Aufgaben durchgeführt:

- Erläuterung der Berichtsszenarien und Herstellen des notwendigen Projektverständnisses bei den Kooperationspartnern
- Unterstützung der Kooperationspartner mit spezifisch auf die einzelnen QSPrüfungen zugeschnittenen Testdaten und zugehörigen Testergebnissen
- Erstellen der P23R SourceConnectoren, die die zu berichtenden Daten im XML-Format bereit stellen
- Erstellen der P23R CommunicationConnectoren, die die von der jeweiligen P23RRegel bereit gestellten Daten an den Datenempfänger übermitteln
- Herbeiführen der Lauffähigkeit eines P23R-Prozessors, der Source- und CommunicationConnectoren im Realeinsatz nutzt
- ichtsempfänger, die unteren Wasserbehörden bzw. das jeweilige Landesumweltamt

D.2.1. Szenarien und Testdaten

Das Szenario für die Übermittlung von Berichtsdaten aus der Selbstüberwachung von Abwasseranlagen gemäß §61 WHG – in diesem Projekt werden Kläranlagen und insbesondere der Kläranalgenjahresbericht betrachtet, nicht jedoch das Kanalnetz und Regenrückhalteeinrichtungen – wird neu entwickelt. Der Berichtsprozess (Abb. D.5) ist hier in der Notation BPMN dokumentiert, die von Fraunhofer FOKUS und Atos für die Zwecke des P23R favorisiert wird.

Für die Abstimmung der Entwicklungsaufgaben zwischen den verschiedenen Auftragnehmern wird die Dokumentation der Anwendungsarchitektur als Komponentendiagramm (Abb. D.6) erstellt.

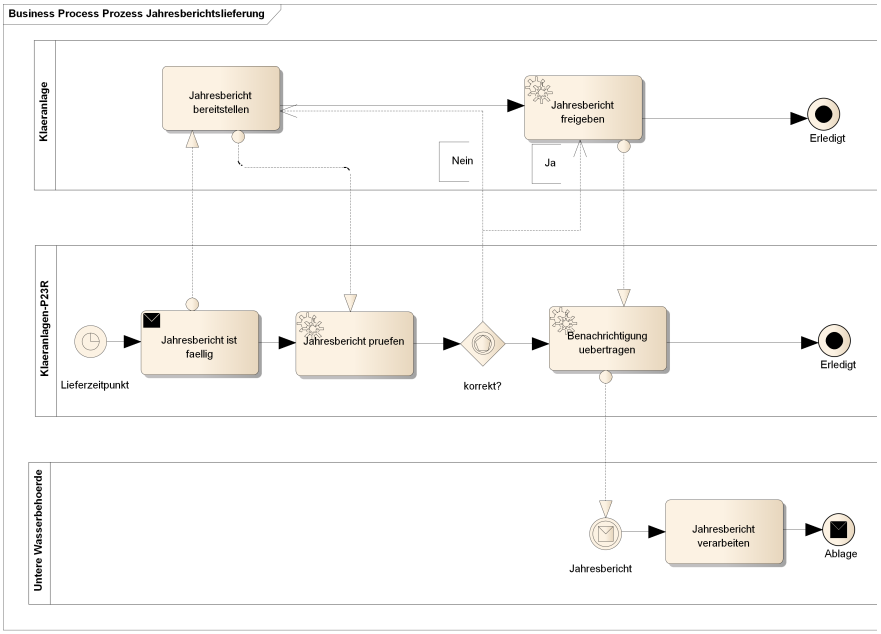


Abbildung D.5.: P23R Geschäftsprozess „Jahresberichtslieferung gem. §61 WHG“

Um dem Berichtslieferanten eine strukturierte Vorgehensweise bei der Verfolgung von Berichtsmängeln zu bieten, werden die möglichen fachlichen Berichtsfehler bzw. warnungen in gleicher Weise gegliedert wie die Prüfungen. Dazu wird jeder Fehlersituation ein aussagekräftiger Name und eine Beschreibung zugeordnet. Die Position des Fehlers wird durch die Benennung des Objekts mit der fehlerhaften Information übermittelt. Dieses kann der Bearbeiter i.d.R. (dies ist von der jeweils verwendeten IT-Lösung abhängig) sofort lokalisieren. Eine Zeilennummer ist bei automatisiert erstellten Dateien – was bei der Übermittlung von XML-Daten der Regelfall ist – nicht hilfreich für die Lokalisierung eines fachinhaltlichen Fehlers.

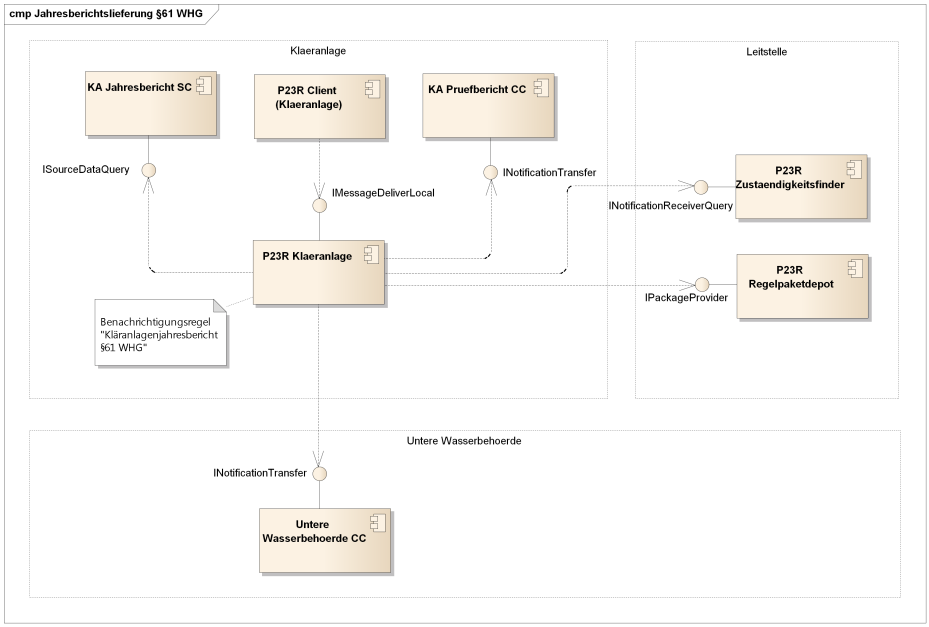


Abbildung D.6.: P23R Komponentendiagramm „Kläranlagenjahresbericht gem. §61 WHG“

D.2.2. Connectoren

Die Entwicklung sowohl der SourceConnectoren als auch der CommunicationConnectoren wird hier unter Einsatz der Skriptsprache Perl durchgeführt.
Die §61-Berichterstattung benötigt keine Objekthistorie und kommt daher mit einem SourceConnector (SC3) aus. Im Übertragungsprozess §61 WHG ist vorgesehen, dass der Datenlieferant in jedem Fall und der Datenempfänger in keinem Fall das Prüfprotokoll erhält. Im Fall einer Prüfung ohne erkennbare Qualitätsmängel der Daten erhält der Berichtsempfänger die für ihn bestimmten Daten. Treten bei der Prüfung Qualitätsmängel an den Daten zutage, erhält der Datenlieferant die Daten zusammen mit dem Prüfprotokoll zurück. Gemäß dem P23R-Prinzip werden diese Aufgaben je Berichtspflicht jeweils von zwei CommunicationConnectoren (CC) durchgeführt, die sich in ihrem internen Aufbau jedoch nur wenig unterscheiden.

D.2.3. Datenmodell für den Testfall §61 WHG

Das Land Bayern hat sich bereit erklärt, dem Forschungsnehmer den elektronisch vorliegenden Datenmodellentwurf des neuen bayerischen Abwasserinformationssystems (Da-BAY) zur Verfügung zu stellen. Daraus werden die für die §61-Berichterstattung in Bayern verwendeten Modellteile – im wesentlichen also der Kläranlagenjahresbericht – extrahiert und mit XUKommunalabwasser zu einem Datenmodell „XUKA-Selbstüberwachung“ zusammengefügt[1].
Die bereits für die Kommunalabwasserberichterstattung erstellten Modellteile werden soweit möglich wiederverwendet. Neue Teile des Datenmodells werden angepasst, so

[1] http://xml.xubetrieb.de/xuka_selbstueberwachung/ver1.0.1/xuka_ selbstueberwachung_nachrichten.html

dass sie konform zu XUBetrieb sind.
Es ist festzustellen, dass der Grad der Überdeckung zwischen den Datenmodellen XUKommunalabwasser und Kläranlagenjahresbericht entgegen der ursprünglichen Annahme sehr gering ist, obwohl die semantischen Räume recht ähnlich sind. Die Typisierung der einzelnen Attribute weicht erheblich von einander ab. Die Struktur der Datenmodelle ist völlig unterschiedlich.
Das für den Test der P23R-konformen Umsetzung neu entwickelte Pivotmodell wurde eingehend untersucht, um Prüfungen zur Sicherung der fachlichen Datenqualität zu entwickeln. Diese werden in Kap. D2.4 vorgestellt.

D.2.4. QS-Prüfungen bei Übermittlung von Selbstüberwachungsdaten nach §61 WHG

Bei der Übermittlung von Berichtsdaten nach §61 Wasserhaushaltsgesetz (WHG) handelt es sich um die Daten der Eigenüberwachung von Abwasseranlagen. Diese umfassen Kläranlagen, Kanalnetz und Regenrückhalteeinrichtungen. Dabei sind für die Prozesskette von der Anlage bis zur Europäischen Kommission (KOM) nur die Berichtsdaten der Kläranlagen von Interesse, da diese in die Berichterstattung zur EU-Kommunalabwasser einfließen.
Die Prüfungen zur Qualitätssicherung (QS-Prüfung) der Daten waren dafür vollständig neu zu entwickeln, da solche bisher nicht vorlagen. Die im Rahmen dieses Vorhabens erarbeiteten Prüfungen gehen stellenweise über die Komplexität der aus der Kommunalabwasserberichterstattung bekannten Prüfungen hinaus und wurden daher nicht in der Sprache OCL notiert, sondern verbal und in Form von Gleichungen unter Verwendung von XPath-Ausdrücken zur Identifikation der zu vergleichenden Werte beschrieben.
Bezüglich der Kategorisierung von QS-Prüfungen wird auf das Kapitel C2 „Detaillierte Beschreibung der QS-Prüfungen bei Übermittung von Kommunalabwasserberichtsdaten“ verwiesen.
Als Referenz für die Entwicklung von QS-Prüfungen für die Berichtsdaten gem. §61 WHG wurde das neu entwickelte Datenmodell „XUKA-Selbstüberwachung“ (siehe Anhang D.2.6) verwendet.

D.2.4.1. Detaillierte Beschreibung der QS-Anforderungen

Die hier aufgeführten Prüfbedingungen sind in positiv-Logik formuliert, müssen folglich eingehalten werden, um einen gültigen Datensatz zu erhalten. Ist zur Auswirkung der Verletzung einer Bedingung nichts weiter angegeben, so ist die Verletzung der Bedingung ein Fehler und führt zur Zurückweisung des Datensatzes. Eine exakte Gleichheit der numerischen Größen ist in der Praxis oft nicht zu erreichen, weshalb bei Vergleichen von Dezimalzahlen meist eine Toleranz von 0,1% vorgesehen wurde. Auf diese Toleranz wurde verzichtet, wenn Werte miteinander verglichen werden, die in der Praxis deutlich von einander abweichen (z. B. Jahresabwassermenge und Jahresschmutzwassermenge) und der Vergleich daher einer Toleranz nicht bedarf.
Der XPath

bericht/klaeranlage/jahresberichtMessstelle/messstelle/betriebszustaende[Normalbetrieb]/abwasserdurchflussKommunal wird im folgenden Text abgekürzt als .../abwasserdurchflussKommunal.

Der XPath bericht/klaeranlage/jahresberichtMessstelle/messstelle/ betriebszustaende[Normalbetrieb]/ betriebszustandParameter wird im folgenden Text abgekürzt als .../betriebszustandParameter.
Der Umfang der ermittelten Prüfungen ist größer, als ursprünglich bei der Befragung des Kooperationspartners und der Befragung dessen Implementierers zu erwarten war. Um die verfügbaren Projektmittel optimal zu nutzen, sind für den Nachweis der Funktionsfähigkeit von QS-Prüfungen im Rahmen des P23R-Prinzips verzichtbare Prüfungen mit dem Symbol ø gekennzeichnet. Diese werden im Rahmen der aktuellen Projektphase nicht implementiert.

Strukturelle Prüfungen, Teil 1 Strukturelle Prüfungen, Teil 1 umfasst die Prüfung abhängiger Pflichtfelder sowie von Listenwerten aus Listen, deren Umfang den Einsatz von XML Schema zur Validierung ausschließt.
§61 WHG kennt keine Listen, deren Umfang die Verwendung in einem XML Schema ausschließt. Diese Prüfungskategorie entfällt hier. Folgende Prüfungsvorschriften für abhängige Pflichtfelder konnten identifiziert werden:
Ob der Wert eines Parameters X (meist Konzentration) an einer Messstelle in einem Betriebszustand in einem Monat ermittelt wurde, kann daran erkannt werden, dass Durchschnitts-, Minimal- oder Maximalwert angegeben wurden.

- Es existieren .../betriebszustandParameter[parameter=X]/konzentration/konzentrationAvg/monat_xWert
- oder .../betriebszustandParameter[parameter=X]/konzentration/konzentrationMax/monat_xWert
- oder .../betriebszustandParameter[parameter=X]/konzentration/konzentrationMin/monat_xWert

Dann müssen alle drei Werte (...Avg, ...Max und ...Min) vorliegen, also jeweils auch die beiden anderen, wenn einer gefunden wurde (Nr. 1.1). Außerdem müssen dann auch alle anderen Werte unterhalb von .../betriebszustandParameter[parameter=X]/konzentration vorhanden sein (Nr. 1.2, ø):
•.../betriebszustandParameter[parameter=X]/konzentration/anzahlMessungen/monat_xWert
25
•.../betriebszustandParameter[parameter=X]/konzentration/anzahlUeberschreitungen/monat_xWert
•.../betriebszustandParameter[parameter=X]/konzentration/summeFrachtenAblauf/monat_xWert
•.../betriebszustandParameter[parameter=X]/konzentration/anzahlFrachtenAblauf/monat_xWert
•.../betriebszustandParameter[parameter=X]/konzentration/eliminationsleistungProzent/monat_xWert

- Wenn außerdem bericht/klaeranlage/reinigtKommunalesAbwasser=true ist, dann muss es an der Messstelle im betrachteten Betriebszustand und unabhängig vom betrachteten Parameter für den Monat, in dem ein Parameter gemessen wurde, auch die folgenden Monatswerte geben (Nr. 1.3): .../abwasserdurchflussKommunal/<alle Elemente>/monat_xWert mit <alle Elemente> jeweils einem der folgenden Elemente:
  - trockenwettertage
  - schmutzwasserdurchflussM3[2] ø

[2] Die XML-Elementnamen und ihre Suffixe „M3“ (m^{3}, Kubikmeter) und M3H (m^{3}\text{/h}, Kubikmeter pro Stunde) wurden dem §61-Datenmodell entnommen.

- abwasserdurchflussM3 ø
- maxTrockenwetterdurchflussM3H ø
- trockenwetterdurchflussAnzahlMessungen ø
- trockenwetterdurchflussAnzahlUeberschreitungen ø
- maxTagestrockenwetterdurchflussM3H ø
- tagestrockenwetterdurchflussAnzahlMessungen ø
- tagestrockenwetterdurchflussAnzahlUeberschreitungen ø
- maxMischwasserdurchflussM3H ø
- mischwasserdurchflussAnzahlMessungen ø
- mischwasserdurchflussAnzahlUeberschreitungen ø
- fremdwasseranteilProzent ø
- anzahlMessungenFremdwasseranteil ø

Strukturelle Prüfungen, Teil 2 Der zweite Teil der strukturellen Prüfungen betrachtet Multiplizitäten berichteter Objekte untereinander unter Berücksichtigung der Objektstatus:
Die Datenstruktur der Nachricht Jahresbericht §61 WHG ist streng hierarchisch und benötigt keine über die Möglichkeiten von XML Schema hinausgehende Überwachung von Multiplizitäten.

Inhaltliche Prüfungen Inhaltliche Prüfungen betrachten den Umfang der berichteten Objekte, wozu Zugriff auf Daten der Vorperiode benötigt wird.
Diese Art von Prüfungen ist speziell für die Anforderungen der europäischen Kommission entwickelt worden, um eine lückenlose Überwachung einer großen Zahl von Objekten über einen längeren Zeitraum zu gewährleisten. Da die Selbstüberwachung nach §61 WHG die Daten nur jeweils einer Kläranlage übermittelt, sind derartige Prüfungen nicht angebracht und daher im beauftragten Szenario auch nicht vorgesehen.

Plausibilität objekintern Die Jahresabwassermenge setzt sich aus der Jahresschmutzwassermenge, dem Regenwasser und evtl. Fremdwasser zusammen. Daher muss gelten (Nr. 4.1):

- bericht/klaeranlage/jahresberichtMessstelle/jahresabwassermengeGesamt >= bericht/klaeranlage/jahresberichtMessstelle/jahresschmutzwassermengeGesamt * 0.999

Die Anzahl der Trockenwettertage in einem Monat muss kleiner oder gleich der Anzahl der Tage des betreffenden Monats sein (Nr. 4.2, ø):

- .../abwasserdurchflussKommunal/trockenwettertage/monat_xWert <= (Anz. Tage im monat_x)

Die Summe der Trockenwettertage über alle gemeldeten Monate muss gleich dem Jahreswert der Trockenwettertage sein (Nr. 4.3):

- Summe(.../abwasserdurchflussKommunal/trockenwettertage/januarWert .. dezemberWert) = .../abwasserdurchflussKommunal/trockenwettertage/jahresWert

Der an einer Messstelle in einem Betriebszustand gemessene Schmutzwasserdurchfluss-Jahreswert muss kleiner oder gleich der gesamten Jahresschmutzwassermenge im Berichtszeitraum sein (Nr. 4.4):

- .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/jahresWert <= jahresberichtMessstelle/jahresschmutzwassermengeGesamt * 1.001

Weiterhin muss der an einer Messstelle in einem Betriebszustand gemessene Schmutzwasserdurchfluss-Jahreswert der Summe der monatlichen Schmutzwassermengen gleichen (Nr. 4.5):

- .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/jahresWert <= Summe(.../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/januarWert .. dezemberWert) * 1.001 AND .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3 /jahresWert >= Summe(.../abwasserdurchflussKommunal/schmutzwasserdurchflussM3 \ /januarWert .. dezemberWert) * 0.999

Analoge Bedingungen gelten für den Abwasserdurchfluss (Nr. 4.6, ø):
Der an einer Messstelle in einem Betriebszustand gemessene Abwasserdurchfluss-Jahreswert muss kleiner oder gleich der gesamten Jahresabwassermenge im Berichtszeitraum sein:

- .../abwasserdurchflussKommunal/abwasserdurchflussM3/jahresWert <= jahresberichtMessstelle/jahresabwassermengeGesamt * 1.001

Weiterhin muss der an einer Messstelle in einem Betriebszustand gemessene Abwasserdurchfluss-Jahreswert der Summe der monatlichen Abwassermengen gleichen (Nr. 4.7, ø):

- .../abwasserdurchflussKommunal/abwasserdurchflussM3/jahresWert <= Summe(.../abwasserdurchflussKommunal/abwasserdurchflussM3/januarWert .. dezemberWert) * 1.001 AND .../abwasserdurchflussKommunal/abwasserdurchflussM3/jahresWert >= Summe(.../abwasserdurchflussKommunal/abwasserdurchflussM3/januarWert .. dezemberWert) * 0.999

Darüber hinaus muss auch hier die Beziehung Abwassermenge >= Schmutzwassermenge gelten. Dies gilt für den Jahreswert (ohne Toleranz, da mindestens ein Regentag im Jahr vorausgesetzt werden kann) aber auch für die Monatswerte (mit Toleranz, da ein Monat ohne Regen vorkommen kann):

- (Nr. 4.8.1) .../abwasserdurchflussKommunal/abwasserdurchflussM3/jahresWert >= .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/jahresWert

- (Nr. 4.8.2) .../abwasserdurchflussKommunal/abwasserdurchflussM3/monat_xWert >= .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3 \ /monat_xWert * 0.999

Beim Trockenwetterdurchfluss wird zwischen stündlichen Maxima (maxTrockenwetterdurchflussM3H in m^3/h) und Tagesmaxima (maxTagestrockenwetterdurchflussM3H auch in m^3/h) unterschieden. **Achtung**: Die Elementnamen sind sehr ähnlich. Das in einem Monat gemessene Stundenmaximum des Trockenwetterdurchflusses muss

größer oder gleich dem aus dem Schmutzwasserdurchfluss (der keine Stromgröße, sondern ein Volumen ist) errechneten mittleren Schmutzwasserdurchfluss sein, denn die Schmutzwassermenge wird aus dem Durchfluss bei Trockenwetter errechnet. Es muss folglich gelten (Nr. 4.9.1):

- .../abwasserdurchflussKommunal/maxTrockenwetterdurchflussM3H/monat_xWert >= .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/monat_xWert / (24.0 * (Anz. Tage in monat_x))

Weiterhin sollte das über einen Zeitraum einer Stunde ermittelte Maximum eines Volumenstroms größer oder gleich dem über den Zeitraum eines Tages (24h) ermittelte Maximum des selben Volumenstroms sein (beide werden in m^3/h angegeben) (Nr. 4.9.2):

- .../abwasserdurchflussKommunal/maxTrockenwetterdurchflussM3H /monat_xWert >= .../abwasserdurchflussKommunal/maxTagestrockenwetterdurchflussM3H /monat_xWert * 0.999

Die über den Zeitraum eines Monats beobachteten maximalen Volumenstöme auf stündlicher Basis sollten nicht größer sein als der über den Zeitraum eines Jahres beobachtete maximale Volumenstrom auf stündlicher Basis. Hier die Bedingung für den Trockenwetterdurchfluss (stündliche Maxima) (Nr. 4.10, ø):

- .../abwasserdurchflussKommunal/maxTrockenwetterdurchflussM3H/monat_xWert <= .../abwasserdurchflussKommunal/maxTrockenwetterdurchflussM3H/jahresWert * 1.001

Auch das Maximum des Tagestrockenwetterdurchflusses muss größer als der aus der Schmutzwassermenge errechnete Mittelwert sein (Nr. 4.11, ø):

- .../abwasserdurchflussKommunal/maxTagestrockenwetterdurchflussM3H/monat_xWert >= .../abwasserdurchflussKommunal/schmutzwasserdurchflussM3/monat_xWert / (24.0 * Anz. Tage in monat_x) Divisor 24.0 kommt dazu, da der maximale Tagestrockenwetterdurchfluss in m^3/h und nicht in m^3/d angegeben wird.

Eine zu Trockenwetterdurchfluss und Schmutzwassermenge analoge Beziehung besteht bei Mischwasserdurchfluss und Abwassermenge. Die Abwassermenge bildet sich aus dem Mischwasserdurchfluss. Es muss folglich gelten (Nr. 4.12.1, ø):

- .../abwasserdurchflussKommunal/maxMischwasserdurchflussM3H/monat_xWert >= .../abwasserdurchflussKommunal/abwasserdurchflussM3/monat_xWert / (24.0 * (Anz. Tage in monat_x))

Auch beim Mischwasser gilt: Die über den Zeitraum eines Monats beobachteten maximalen Volumenstöme auf stündlicher Basis sollten nicht größer sein als der über den Zeitraum eines Jahres beobachtete maximale Volumenstrom auf stündlicher Basis (Nr. 4.12.2, ø):

- .../abwasserdurchflussKommunal/maxMischwasserdurchflussM3H /monat_xWert <= .../abwasserdurchflussKommunal/maxMischwasserdurchflussM3H /jahresWert * 1.001

Plausibilität objektübergreifend Objektübergreifende Prüfungen entfallen, da keine nebeneinander stehenden Entitäten im Datenmodell vorhanden sind. Daher sind auch keine auswertbaren Beziehungen solcher Entitäten zu einander vorhanden.

E. Entwicklung wiederverwendbare Bausteine

E.1. Analyse der Prozessketten und Entwicklung wiederverwendbare Bausteine und Regelkomponenten

In einem internen Workshop (FhG, ENDA) wurden erste Ideen zusammengetragen und in einer Mindmap skizziert.

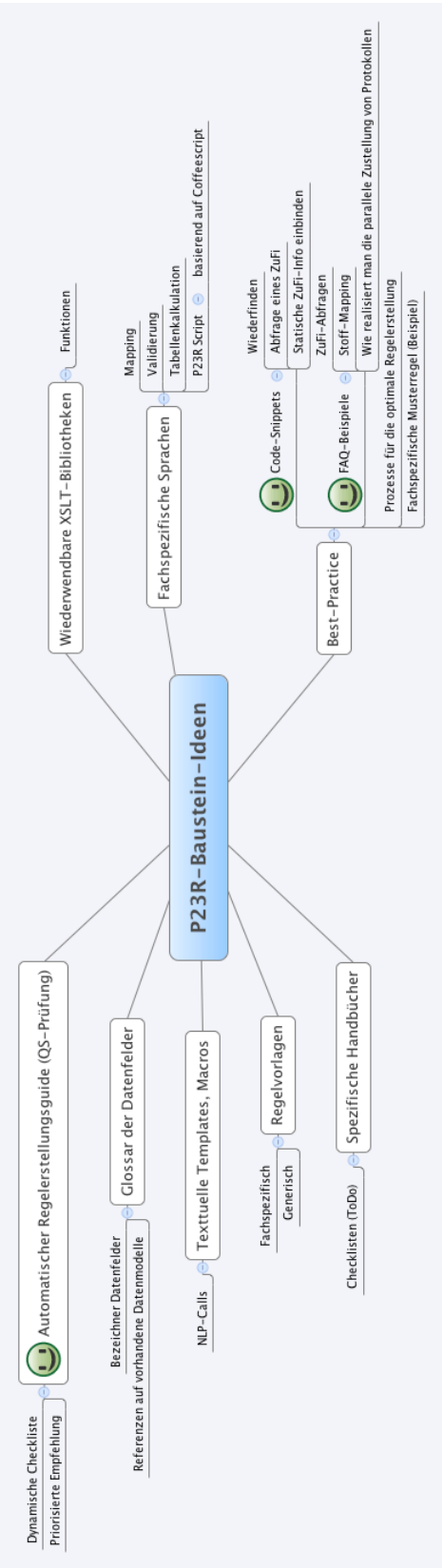


Abbildung E.1.: Mindmap: Ideen zu wiederverwendbaren Bausteinen und Regelkomponenten

Diese Ideen wurden daraufhin überarbeitet und in einem Dokument zusammengefasst. Das folgende Kapitel „Vereinfachte Regelerstellung von P23R-Benachrichtigungsregeln für die Domäne Umwelt" gibt diesen Dokument wieder. In weiteren Projekten der Zukunft ist es sinnvoll, diese Bausteine weiter auszubauen. Besonderes Augenmerk ist dabei auf die FAQs zu legen.

E.2. Vereinfachte Regelerstellung von P23R-Benachrichtigungsregeln für die Domäne Umwelt

Das hier aufgeführte Handbuch zur Regelerstellung wurde von FhG und Atos basierend auf den Ergebnissen eines Workshops mit ENDA erstellt.

E.2.1. Einleitung

Im Rahmen des Projektes P23R4Flex wurden zwei Benachrichtigungsregelgruppen für den Bereich XUKommunalabwasser XUK und §61 WHG erstellt. Das vorliegende Ergebnisdokument soll anhand der erarbeiteten P23R-Benachrichtigungsregelgruppen aufzeigen, ob die Regelerstellung für die Domäne Umwelt vereinfacht werden. Die ursprüngliche These war, dass es umweltspezifische Funktionsbibliotheken geben könnte, die immer wieder benötigte Funktionen in der Domäne Umwelt beinhalten. Im Sinne des wesentlichen Ziels, die Regelerstellung zu vereinfachen und den damit verbunden Entwicklungsaufwand zu minimieren, wurden im Rahmen dieser Arbeit auch Alternativen zu Funktionsbibliotheken betrachtet und untersucht. Das vorliegende Ergebnisdokument fasst die Analyse der erstellten Benachrichtigungsregelgruppen und die Diskussionen aus den Workshops mit allen beteiligten Stakeholder zusammen, insbesondere die der beteiligten Entwickler.
Zuerst wird entsprechend der ursprünglichen Idee zur Reduzierung des Entwicklungsaufwandes die Einsatzmöglichkeit von Funktionsbibliotheken betrachtet. Darauf aufbauend werden Domain Specific Languages (DSL) betrachtet, die auf sprachlicher Ebene den Aufwand zur Erstellung, für die Qualitätssicherung, bspw. durch einfachere Reviews, und zur Wartung durch einfachere, effektivere oder gewohntere Schreibweise reduzieren könnte. Ein pragmatisches aber effektives Mittel stellen verbesserte wiederverwendbare Artefakte dar, beispielsweise die Dokumentation von Best-Practices. Ein weiterer Ansatz ist die Erstellung fachspezifischer Artefakte, deren Verwendung durch vorbereitete und teilweise vorausgefüllte Artefakte einer Benachrichtigungsregel oder -regelgruppe den Einstieg erleichtern und den Aufwand durch die bereits vorhandene Teile reduzieren. Anschließend betrachten wir den Aspekt der verbesserten und der fachspezifischer Dokumentation. Ein wesentlicher Punkt ist auch die Verbesserung der Dokumentation der Benachrichtigungsregeln und Datenmodelle. Am Ende wird auch noch die Idee eines automatischen Regelerstellungsassistenten betrachtet, der den Regelersteller oder Datenmodellierer durch den Erstellungsprozess von Benachrichtigungsregelgruppen und Pilot-Teildatenmodellen zielgerichtet leitet.
Den Abschluss bildet ein Fazit, das Vorschläge enthält welche Maßnahmen mit welcher Priorität umgesetzt werden sollten, um die Regelerstellung und Wartung im Fachgebiet Umwelt zu verbessern, d.h. den notwendigen Aufwand zu reduzieren.

E.2.2. Fachspezifische XSLT-Bibliotheken

Die ursprüngliche Idee war die Erstellung einer für die Domäne Umwelt fachspezifische Funktionsbibliothek in XSLTv2 für P23R-Regelersteller zu konzipieren und ggfs. zu entwickeln. Wiederverwendbare P23R-Bibliotheken sollen in P23R-Benachrichtigungsregeln und -regelgruppen direkt eingebunden werden können. Die Bibliotheken beinhalten eine Menge von wiederverwendbaren Funktionen, die einfach mit unterschiedlichen Parametern im konkreten Anwendungsfall aufgerufen werden können. Die Benachrichtigungsregeln und -gruppen, die in der technischen Benachrichtigungsregelsprache (T-BRS) geschrieben werden, erlauben grundsätzlich die Nutzung von mit der Benachrichtigungsregelgruppe mitgelieferten XSLTv2-Bibliotheken. Sie sollten dabei Funktionen mit immer wiederkehrenden Berechnungen bieten, bei deren Nutzung der übrige Code in den Skripten einer Benachrichtigungsregel, Benachrichtigungsregelgruppe oder eines Testfalls sich merklich reduziert. Es ist konkret zu prüfen, ob in den für P23R4Flex bzw. für das Fachgebiet Umwelt erstellten Benachrichtigungsregeln entsprechend als Funktion wiederverwendbare Code-Teile identifiziert und heraus gelöst werden können, d.h. regelübergreifend von Nutzen wären und daher Teil solch einer Bibliothek sein sollten. Für jeden Einzelfall ist auch zu prüfen, ob es sich wirklich um eine potentielle Funktion oder lediglich um Code-Snippets bzw. Macros handelt!
Die Analyse der für P23R4Flex erstellten Benachrichtigungsregeln ergab, dass keine fachspezifischen Beispiele identifiziert werden können. Ein häufigeres Problem waren Wertvergleiche, bei denen Toleranzen beim Vergleich berücksichtigt werden müssen, bspw. "a equal b" –> "a - b < 0.1". Dies ist allerdings ein sehr allgemeines Problem. Hier könnte man prüfen, ob es bereits "mathematische" XSLTv2-Bibliotheken gibt, deren Einbindung in Benachrichtigungsregelgruppen bei Bedarf empfohlen werden kann, bspw. durch einen entsprechenden Eintrag in den FAQs des P23R-Entwicklerportals.
Die Analyse der Benachrichtigungsregeln ergab jedoch auch den Bedarf an allgemein für P23R-Benachrichtigungsregeln wiederverwendbaren Funktionen für das Logging. Die beiden Benachrichtigungsregeln für P23R4Flex haben die ohne Weiteres allgemein verwendbare Idee und Anforderung, dass die Ergebnisse der syntaktischen und semantischen Validierung der Quelldaten nicht nur im Log des P23R gesammelt und bereit gestellt werden, sondern in Form eines Prüfprotokolls sowohl an den Benachrichtigungssender als auch den Benachrichtigungsempfänger bspw. per E-Mail versendet werden. Dafür wurde der zurzeit teilweise spezielle Code verwendet:

```
...
<xsl:template name="Fehler">
        <xsl:param name="kontext"/>
        <xsl:param name="code"/>
        <xsl:param name="name"/>
        <xsl:param name="beschreibung"/>
        <xsl:param name="position"/>
        <xukn:fehler>
                <xukn:kontext><xsl:value-of select="$kontext"/></xukn:kontext>
                <xukn:zeitpunkt><xsl:value-of select="current-dateTime()"/></xukn:zeitpunkt>
                <xukn:code><xsl:value-of select="$code"/></xukn:code>
                <xukn:name><xsl:value-of select="$name"/></xukn:name>
                <xukn:beschreibung><xsl:value-of select="$beschreibung"/></xukn:beschreibung>
                <xukn:position><xsl:value-of select="$position"/></xukn:position>
        </xukn:fehler>
</xsl:template>

<xsl:template name="Warnung">
        <xsl:param name="kontext"/>
        <xsl:param name="code"/>
        <xsl:param name="name"/>
        <xsl:param name="beschreibung"/>
        <xsl:param name="position"/>
        <xukn:warnung>
                <xukn:kontext><xsl:value-of select="$kontext"/></xukn:kontext>
                <xukn:zeitpunkt><xsl:value-of select="current-dateTime()"/></xukn:zeitpunkt>
                <xukn:code><xsl:value-of select="$code"/></xukn:code>
                <xukn:name><xsl:value-of select="$name"/></xukn:name>
                <xukn:beschreibung><xsl:value-of select="$beschreibung"/></xukn:beschreibung>
                <xukn:position><xsl:value-of select="$position"/></xukn:position>
        </xukn:warnung>
</xsl:template>
...
```

Diese Vorgehensweise von Benachrichtigungsregeln mit einem Validierungsprotokoll ist nicht nur spezifisch für die Domäne Umwelt aber wäre in dieser Domäne ein guter Kandidat für eine Funktionsbibliothek. Daher sollten die bisher erstellten Funktionen verallgemeinert und bei Bedarf im Rahmen einer allgemeinen Verwendung ergänzt werden. Darüber hinaus ist zu prüfen, ob diese eventuell auch als Standardfunktionen für das Logging in die T-BRS einfließen könnten. Solch eine Bibliothek kann auch Teil einer generellen Validierungslösung sein, wie sie in Form einer fachspezifischen Sprache (Domain Specific Language; DSL) für die Validierung bzw. P23R Script bereits angedacht ist.
Ein weiterer Bereich für Funktionsbibliotheken könnte mittelfristig der Umgang mit Maßeinheiten sein, d.h. dass Messwerte immer zusammen mit ihrer Maßeinheit angegeben werden müssen, damit die Dimensionierung einheitlich ist, d.h. alle Werte werden auf eine Grundeinheit normiert: $cm(100) == m(1)$
Entsprechend müssten Funktionsbibliotheken erstellt werden, die die Masseinheiten umrechnen. Dabei sollte dann auch geklärt werden, ob die Nutzung von Masseinheiten mittels Funktionsbibliotheken zu übersichtlichen und wattbaren Code führt oder ob eher andere Wege wie P23R Script mit Masseinheiten lediglich ein gangbarer Weg sind: $if 100cm == 1m then ...$
Beim weiteren Vorgehen sollten zuerst Beispiele erarbeitet werden, um den Nutzen der Funktionsbibliotheken besser einschätzen zu können. Ferner sind auch in Zukunft neue Benachrichtigungsregeln auf Verbesserungsmöglichkeiten durch den Einsatz von Funktionsbibliotheken zu prüfen. Dazu sollte im Info/FAQ-Bereich des P23R-Entwicklerportals eine kommentierte Liste mit potentiellen und empfohlenen XSLTv2-Bibliotheken bereit gestellt und gepflegt werden.

E.2.3. Fachspezifische Sprachen

Fachspezifische Sprachen (DSLs) erlauben es spezifische Probleme kompakter und lesbarer zu beschreiben. Für die Regelersteller ergibt sich durch die kompaktere Beschreibung ein geringerer Aufwand bei der Erstellung aber auch der Wartung der Benachrichtigungsregeln. Für die fachlichen Reviewer von Benachrichtigungsregeln werden diese lesbarer, verständlicher und vor allem nachvollziehbarer. Fachspezifische Sprachen haben immer eine eigene, formale Grammatik und müssen grund-

sätzlich in die technische Benachrichtigungsregelsprache des P23R-Prozessors (T-BRS) übersetzbar sein, bspw. mittels eines Compilers. Man kann fachspezifische Sprachen durch entsprechende Restriktionen und zulässige Sprachmuster an die natürliche Sprache annähern. Ein direktes Verstehen von natürlicher Sprache im Allgemeinen ist technisch bisher nicht möglich, d. h. das korrekte Schreiben von Artefakten in einer fachspezifischen Sprache wird unter Umständen entsprechende Kompetenzen und technische Unterstützung ergänzend erfordern.
Nachfolgend sind mögliche Anwendungen und Lösungsansätze für fachspezifische Sprachen im Kontext Umwelt aufgeführt.

E.2.3.1. Mapping

Im Rahmen des ursprünglichen P23R-Projektes erstellten Benachrichtigungsregeln für Umwelt im Bereich Immissionen mussten Werte massenweise transformiert werden, d. h. von einem Gefahrenstoffkatalog in einen anderen übersetzt werden. Statt dieses Mapping durch eine umfangreiche Anzahl an Fallunterscheidungen in XSLTv2 zu programmieren, könnte das Mapping aber auch in einer Kurzschreibweise aufgeschrieben werden, bspw.:

```
Dieses Mapping beschreibt die Abbildung von Katalog A in B

  Konvertiere x

Eine einfache Konvertierung von Stoffen.

  Stoff 1a -> Stoff 1b
  Stoff 2a -> Stoff 2b
  Stoff 3a -> Stoff 3b

...
```

Komplexere Konvertierungen könnten natürlich auch Kontexte (Tupel von optionalen Werten) oder beliebige Bedingungen beinhalten:

```
Dieses Mapping beschreibt die Abbildung von Katalog A in B

  Konvertiere x im Kontext y,z

  Stoff 1a -> Stoff 1b

Bilde eine Stoffkombination (x,y) ab.

  Stoff 2a, Stoff 101 -> Stoff 2b

Bilde nur ab, wenn ein bestimmter Wert in z vorliegt.

  Stoff 3a, _ , 100 -> Stoff 3b

Bilde nur ab, wenn die Bedingung erfüllt ist.

  Stoff 4a -> Stoff 4b wenn z > 100

...
```

Dieses P23R-Mapping würde dann direkt ein XSLTv2-Skript erzeugen, dass direkt in die Transformationsschritte im Manifest einer Benachrichtigungsregel eingefügt werden kann. Im obigen Beispiel kann das Mapping im Literate-Programming-Style auch direkt kommentiert und dokumentiert werden. Für das einfache Mapping von Werten bietet sich aber auch die Nutzung von Tabellen an, bspw. im von allen Bürokommunikationssystemen leicht zu exportierenden Format CSV. Tabellenkalkulationen sind beliebte Werkzeuge und leicht zu benutzen.

Spalte1	Spalte2	Spalte3	Spalte4	Spalte5
alter Stoff	Nebenstoff	Stoffwert	neuer Stoff	Kommentar
Stoff 1a			Stoff 1b	
Stoff 2a	Stoff 101		Stoff 2b	Bilde eine Stoffkombination (x,y) ab.
Stoff 1a		100	Stoff 3b	Bilde nur ab, wenn ein bestimmter Wert in z vorliegt.

```
alter Stoff; Nebenstoff; Stoffwert; neuer Stoff;Kommentar
```

```
Stoff 1a;;;Stoff 1b;
Stoff 2a;Stoff 101;;Stoff 2b;Bilde eine Stoffkombination (x,y) ab.
Stoff 1a;;100;Stoff 3b;Bilde nur ab, wenn ein bestimmter Wert in z vorliegt.
```

In das Mapping integrierte Berechnungen lassen sich mit Tabellenkalkulationen nicht so einfach abbilden!

E.2.3.2. Validierung

Im Rahmen der Benachrichtigungsregeln für Umwelt im Bereich Wasserhaushalt mussten Berichte auf Validität geprüft werden. Die einfacheren Strukturprüfungen wurden dabei vom XML Schema abgedeckt. Zusammenhänge innerhalb eines Berichts, sowie Abhängigkeiten mit früher gemeldeten Berichten, wurden dagegen von den Benachrichtigungsregeln abgedeckt. Die Realisierung bestand darin, die umfangreichen und komplexeren fachlichen Zusammenhängen in XSLTv2 zu programmieren.
Für die Erstellung der Benachrichtigungsregeln wurden die zu implementierenden Validierungen in der formalen Sprache OCL (Object Constraint Language) geliefert. Ein Beispiel aus dem Projekt XU-Kommunalabwasser:

```
context /berichtspflichten.Landesbericht.001/klaeranlage/aktiv/inland inv
  let assoc = /berichtspflichten.Landesbericht.001/zuordnung[klaeranlage/id=self.id] in
  let siedlung = /berichtspflichten.Landesbericht.001/siedlungsgebiet/aktiv/inland[id=assoc/siedlungsgebiet/id] in
  assoc->collect(
    if count(siedlung)=1
    then anteil*siedlung/siedlungsabwasser/nominalbelastung
    else 0 endif
  )->sum() <= 110*ausbaugroesse
```

Die entsprechende Übersetzung in XSLTv2:

```
<xsl:for-each select="$lb/xuk:klaeranlage/xuk:aktiv/xuk:inland">
  <xsl:variable name="id_ctx" select="xuk:id"/>
  <xsl:variable name="name_ctx" select="xuk:name"/>
  <xsl:variable name="collect">
    <xsl:for-each select="$lb/xuk:zuordnung[xuk:klaeranlage/xuk:id = $id_ctx]">
      <xsl:variable name="id1" select="xuk:siedlungsgebiet/xuk:id"/>
      <xsl:variable name="siedlung" select="$lb/xuk:siedlungsgebiet/xuk:aktiv/xuk:inland[xuk:id = $id1]"/>
      <xsl:if test="count($siedlung) = 1">
        <siedlungsgebiet>
          <name><xsl:value-of select="$siedlung/xuk:id"/>, <xsl:value-of select="$siedlung/xuk:name"/></name>
          <value><xsl:value-of select="xuk:anteil * $siedlung/xuk:siedlungsabwasser/xuk:nominalbelastung/xuk:content"/></value>
        </siedlungsgebiet>
      </xsl:if>
    </xsl:for-each>
  </xsl:variable>
  <xsl:variable name="summe" select="sum($collect/siedlungsgebiet/value)"/>
  <xsl:if test="$summe > (110 * xuk:ausbaugroesse)">
    <xsl:call-template name="Warnung">
      <xsl:with-param name="kontext">Kläranlage und zugeordnete Siedlungsgebiete</xsl:with-param>
      <xsl:with-param name="code">...</xsl:with-param>
      <xsl:with-param name="name">...</xsl:with-param>
      <xsl:with-param name="beschreibung">...</xsl:with-param>
      <xsl:with-param name="position">Kläranlage <xsl:value-of select="$id_ctx"/> ...</xsl:with-param>
    </xsl:call-template>
  </xsl:if>
</xsl:for-each>
```

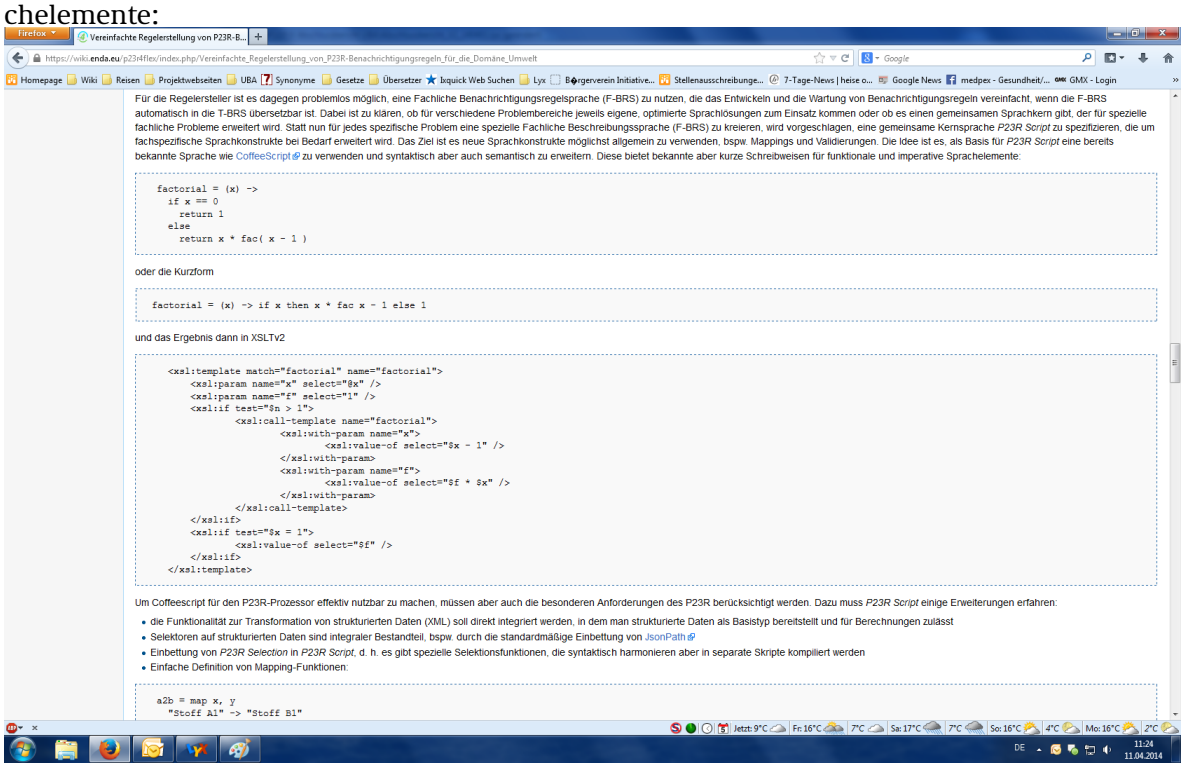
Es ergeben sich mehrere Vorteile, wenn die Validierenden in OCL formuliert werden. OCL ist ein OMG Standard und wohl spezifiziert. Da in OCL formulierte Bedingun-

gen eine formale, ausführbare Spezifikation darstellen, ist die Definition eindeutig und erlaubt keine (Fehl)Interpretationen. OCL-Ausdrücke können bei entsprechendem Vorwissen auch von Fachabteilungen verstanden und unter Umständen sogar erstellt werden. Die Beschreibungen in OCL sind auch deutlich kompakter und lesbarer. Damit sollte sich der Aufwand zur Erstellung und Wartung der Validierungen verringern.
Das P23R-Konzept ermöglicht es spezielle fachspezifische Sprachen für die Regelersteller zu unterstützen, wenn diese in Skripte für die Technische Benachrichtigungsregelsprache automatisch übersetzt werden können. Wesentliche Teile von OCL (Invarianten) können zur Validierung von Daten in XSLTv2 automatisch übersetzt werden. Damit könnten Validierungen in OCL geschrieben und direkt in eine Benachrichtigungsregel als Transformationsschritt integriert werden. Die Übersetzung der OCL-Definitionen kann beispielsweise transparent im P23R-Entwicklerportal passieren.
Wenn man die Validierungen grundsätzlich mit OCL im Kontext P23R definieren und schreiben würde, müsste festgelegt werden, welche Teilmenge von OCL im Kontext P23R wirklich notwendig ist und unterstützt wird. Ferner sollte eine Übersetzung von OCL for P23R nicht nur den eigentlichen Validierungscode in XSLTv2 erzeugen, sondern zusätzlich auch den entsprechenden Code für die Log-Einträge, die im Test-P23R angezeigt werden, und das zu versendende Validierungsprotokoll erzeugen, wie es im Bereich Umwelt benötigt wird.
Grundsätzlich ist aber auch zu klären, ob OCL in seiner bisherigen Form (Syntax) eingesetzt werden soll, oder ob es mit den anderen fachspezifischen Sprachen harmonisiert wird, bspw. mit P23R Selection oder P23R Script. Dabei würde die Funktionalität grundsätzlich erhalten bleiben, sich aber die Schreibweise ändern. Sicherlich kann man für vorhandenen OCL-Code einen Übersetzer bereit stellen, der die OCL-Syntax automatisch in die harmonisierte Form überführt.

E.2.3.3. P23R Script

Die Entwicklung von Programmiersprachen zeigt, dass universelle Programmiersprachen eine hohe Akzeptanz bei Entwicklern fördern. Ferner bevorzugen Entwickler häufig Sprachkonstrukte, die sie bereits kennen und gewohnt sind. Somit ist die Akzeptanz neuer formalen Sprachen stets problematisch. Das Erlernen neuer Sprachen stellt meistens ein größeres Hindernis dar, da insbesondere am Anfang die Effektivität des Entwicklers leidet. Auch die Sprache XSLTv2 ist bei Entwicklern wenig beliebt, da sie durch ihre Schreibweise mit Tags sehr ungewohnt und auch unübersichtlich ist. Entsprechend gab es von Beginn an die Forderung eine eher Java-ähnliche Sprache zur Verfügung zu haben. Die Auswahl von XSLTv2 als Basis für die Technische Benachrichtigungssprache (T-BRS) orientierte sich aber in erster Linie an den technischen Rahmenbedingungen, d. h. eine herstellerneutrale und verbreitete Lösung zur Transformation von strukturierten Daten (XML) ohne großen Aufwand nutzen und einsetzen zu können. Von Anfang an war aber klar, dass dies die technische Sicht des P23R-Prozessors war und ist.
Für die Regelersteller ist es dagegen problemlos möglich, eine Fachliche Benachrichtigungsregelsprache (F-BRS) zu nutzen, die das Entwickeln und die Wartung von Benachrichtigungsregeln vereinfacht, wenn die F-BRS automatisch in die T-BRS über-

setzbar ist. Dabei ist zu klären, ob für verschiedene Problembereiche jeweils eigene, optimierte Sprachlösungen zum Einsatz kommen oder ob es einen gemeinsamen Sprachkern gibt, der für spezielle fachliche Probleme erweitert wird. Statt nun für jedes spezifische Problem eine spezielle Fachliche Beschreibungssprache (F-BRS) zu kreieren, wird vorgeschlagen, eine gemeinsame Kernsprache P23R Script zu spezifizieren, die um fachspezifische Sprachkonstrukte bei Bedarf erweitert wird. Das Ziel ist es neue Sprachkonstrukte möglichst allgemein zu verwenden, bspw. Mappings und Validierungen. Die Idee ist es, als Basis für P23R Script eine bereits bekannte Sprache wie CoffeeScript zu verwenden und syntaktisch aber auch semantisch zu erweitern. Diese bietet bekannte aber kurze Schreibweisen für funktionale und imperative Sprachelemente:



oder die Kurzform

```
factorial = (x) -> if x then x * fac x - 1 else 1
```

und das Ergebnis dann in XSLTv2

```
<xsl:template match="factorial" name="factorial">
    <xsl:param name="x" select="@x" />
    <xsl:param name="f" select="1" />
    <xsl:if test="$n > 1">
            <xsl:call-template name="factorial">
                    <xsl:with-param name="x">
                            <xsl:value-of select="$x - 1" />
                    </xsl:with-param>
                    <xsl:with-param name="f">
                            <xsl:value-of select="$f * $x" />
                    </xsl:with-param>
            </xsl:call-template>
    </xsl:if>
    <xsl:if test="$x = 1">
            <xsl:value-of select="$f" />
    </xsl:if>
</xsl:template>
```

Um Coffeescript für den P23R-Prozessor effektiv nutzbar zu machen, müssen aber auch die besonderen Anforderungen des P23R berücksichtigt werden. Dazu muss P23R Script einige Erweiterungen erfahren:

- die Funktionalität zur Transformation von strukturierten Daten (XML) soll direkt integriert werden, in dem man strukturierte Daten als Basistyp bereitstellt und für Berechnungen zulässt
- Selektoren auf strukturierten Daten sind integraler Bestandteil, bspw. durch die standardmäßige Einbettung von JsonPath
- Einbettung von P23R Selection in P23R Script, d. h. es gibt spezielle Selektionsfunktionen, die syntaktisch harmonieren aber in separate Skripte kompiliert werden
- Einfache Definition von Mapping-Funktionen:

```
a2b = map x, y
  "Stoff A1" -> "Stoff B1"
  "Stoff A2" -> "Stoff B2" when y > 100
  else
     "Stoff C"
```

- Validierung der Werte von Strukturen auf der Basis eines wohldefinierten Subsets von OCL in Coffeescript-Schreibweise
- Gruppierung der Validierungen analog bei Unit-Tests basierend auf der Schreibweise von Behaviour Driven Development (BDD) und

```
describe "Bei aktiven inländische Kläranlagen", berichtspflichten.klaeranlage.aktiv.inland ->

  assoc = berichtspflichten.zuordnung[klaeranlage/id=self.id]
  siedlungen = berichtspflichten.siedlungsgebiet.aktiv.inland[id=assoc/siedlungsgebiet/id]

  it "sollte die Summe aller Abwässer nicht größer als 110% im Verhältnis zur Ausbaugröße sein", ->
    nominalbelastungen = for assoc
      if siedlungen.count() == 1
        anteil * siedlungen.siedlungsabwasser.nominalbelastung
      else
        0
    expect nominalbelastungen.sum() not toBeGreaterThan 110 * ausbaugroesse
```

- Unterstützung von Satzpattern, um Fakten auch natürlich-sprachlich in einem eingeschränkten Deutsch ausdrücken zu können.

Die Realisierung einer einheitlichen Sprachbeschreibung ist technisch machbar, aber auch ambitioniert. Daher würde man bei der Umsetzung einer fachspezifischen Sprache wie P23R Script schrittweise vorgehen. Im ersten Schritt würden einzelne Beispiele erstellen und eine einheitliche Sprachbeschreibung entstehen, bspw. analog dem Vorgehen auf der Website von Coffeescript, wobei die Zielsprache dann gleich XSLTv2 wäre. Als realistischen Proof-of-Concept empfiehlt es sich auch, die vorhandenen Benachrichtigungsregeln für XUK und $61 WHG beispielhaft komplett in P23R Script umzuschreiben, um einen echten Vergleich zu haben. Ein Review mit allen Stakeholder bei solch einem Meilenstein wäre hilfreich, um zu klären ob die Idee praktikabel und nützlich ist, d.h. der Aufwand zum Erstellen von Benachrichtigungsregeln und deren Wartung von P23R Script nachhaltig und merklich verringert wird. Bei der Implementierung von P23R Script würde man dann ebenfalls bedarfsgetrieben vorgehen und beispielsweise erst einmal nur die Validierungen realisieren bevor man den vollen Sprachumfang à la Coffeescript umsetzt. Die Realisierung der Validierungen würde auch ein Übersetzer eines Subsets von OCL in das entsprechende Subset von P23R Script umfassen.
Zusammenfassend ist das Ziel von P23R Script ein für den Regelersteller leichter zu erstellender und wartender Code. Dazu sollen die verwendeten Schreibweisen kompakter und für typische Java/C#/Python/...-Entwickler gewohnter werden. Gleichzeitig sollen auch Nicht-Techniker den Code leichter lesen und verstehen können, bspw. für Reviews und zur Freigabe.

E.2.4. Best-Practices

Best-Practices dokumentieren Erfahrungen, die leicht auf andere Anwendungsfälle - insbesondere Benachrichtigungsregeln und Pivot-Teildatenmodelle - übertragen werden können, bspw. auch durch einfache Anpassung von existierenden Quellcode. Best-Practices können sich auf ein einzelnes Fachgebiet beziehen oder generelle technische Tipps für die Erstellung von Benachrichtigungsregelgruppen und Pilot-Teildatenmodelle sein. Die Idee ist es, dass diese Best Practices im P23R-Entwicklerportal von Regelerstellern und Datenmodellierern für andere selbst bereit gestellt werden können. Nachfolgend sind verschiedene Formen von Best-Practices aufgeführt, die man in unterschiedlicher Form in das P23R-Entwicklerportal optimiert integrieren könnte.

E.2.4.1. Code-Snippets

Die Code-Snippets sind kleine Textschnipsel, die mittels Copy & Paste in Skripte eingefügt und bei Bedarf geringfügig angepasst werden können. Die Textschnipsel eignen sich nicht direkt zur Realisierung mittels einer Funktion oder als Teil einer Bibliothek. Code-Snippets können einerseits im P23R-Entwicklerportal angeboten werden, oder andererseits direkt als spezielles Werkzeug innerhalb eines Code-Editors zum Erstellen von Skripten für Benachrichtigungsregeln. Ein wichtiger Aspekt bei der Umsetzung ist dabei, wie man effektiv das passende Code-Snippet finden kann, bspw. durch Schlag- oder Stichwortsuche. Mögliche Beispiele sind:

- Statische Zuständigkeitsinformationen für typische Meldungsempfänger in eine Benachrichtigungsregel einbinden, bspw. Bundes- und Landesumweltämter
- Aufruf des Zuständigkeitsfinder für typische Meldungsempfänger im Umfeld Umwelt
- Eine Benachrichtigung komplett kopieren:

```
<xsl:template match="/">
  <xsl:apply-templates select="/ng:notificationGeneration/ng:notification"/>
</xsl:template>

<xsl:template match="/ng:notificationGeneration/ng:notification">
  <notification:notification>
    <xsl:apply-templates select="@*|node()" mode="copy"/>
  </notification:notification>
</xsl:template>

<xsl:template match="@*|node()" mode="copy">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()" mode="copy"/>
  </xsl:copy>
</xsl:template>
```

Code-Snippets könnten mit Parametern in Form von Macros noch flexibler gestaltet werden, d. h. die Macros enthalten Platzhalter, die beim Einfügen in das Artefakt einer Benachrichtigungsregel, Benachrichtigungsregelgruppe oder einem Pilot-Teildatenmodells textuell ersetzt werden. Dies wäre vor allem in einem Code-Editor ein nützliches Werkzeug.

E.2.4.2. FAQs

Die Frequently Asked Questions (FAQ) bilden einen wenig strukturierten Wissensspeicher, um Erfahrungen von Regelerstellern und Datenmodellierer an Andere weiter zu geben. Das P23R-Entwicklerportal verfügt bereits über einen öffentlichen Info-Bereich, in dem auch FAQs verfügbar sind. Dieser sollte durch Regelersteller und Datenmodellierer dahin gehend erweiterbar sein, dass diese selbständig für ein Fachgebiet Code-Beispiele hinzufügen können. Mögliche Beispiele für FAQs sind:

- Wie fragt man die Zuständigkeitsinformationen beim Öffentlichen Datenquellkonnektor der P23R-Leitstelle ab?
- Wie integriert man statische Zuständigkeitsinformationen in eine Benachrichtigungsregel?

Die statischen Informationen sollte man grundsätzlich in eine externe XML-Datei auslagern, die Teil der Benachrichtigungsregel oder Benachrichtigungsregelgruppe ist, und im Unterverzeichnis shared liegen muss. Hier ein Beispiel für die Landesumweltämter:

```
<receiver>

  <bundesland id="01" kuerzel="SH" name="Schleswig-Holstein"
              idorg="p23r:org:de.land:LUA_SH" ideservice="p23r:eservice:de.land.LUA_SH:mail"
              email="sh@reply.ka.p23r.enda.eu"/>

  <bundesland id="02" kuerzel="HH" name="Hamburg"
              idorg="p23r:org:de.land:LUA_HH" ideservice="p23r:eservice:de.land.LUA_HH:mail"
              email="hh@reply.ka.p23r.enda.eu"/>

  <bundesland id="03" kuerzel="NI" name="Niedersachsen"
              idorg="p23r:org:de.land:LUA_NI" ideservice="p23r:eservice:de.land.LUA_NI:mail"
              email="ni@reply.ka.p23r.enda.eu"/>

  ...
</receiver>
```

Die Datei mit den statischen Zuständigkeiten kann dann beispielsweise im Skript ProfileTransformation.xslt wie folgt genutzt werden:

```
<xsl:template match="*:notificationProfile/*:communication">

  <communication>

    <publicTimeStamp/>
    <xsl:variable name="rcv"
        select="document('rule://EU_RL/EU_RL_REP/shared/Receiver.xml')/
           receiver/bundesland[@id = $bundesland]"/>

    <xsl:element name="receivers" namespace="{namespace-uri()}">
      <xsl:attribute name="receiverId" select="$rcv/@idorg"/>
      <xsl:attribute name="receiverName" select="concat('Landesumweltamt ', $rcv/@name)"/>
      <xsl:element name="support" namespace="{namespace-uri()}"/>
      <xsl:element name="technicalSupport" namespace="{namespace-uri()}"/>
      <xsl:element name="channels" namespace="{namespace-uri()}">
      <xsl:attribute name="id"></xsl:attribute>
      <xsl:attribute name="name" select="'lua'"/>
      <xsl:attribute name="type" select="$rcv/@ideservice"/>
      <xsl:attribute name="submissionTime">2010-01-01T00:00:00</xsl:attribute>
      <xsl:attribute name="timeoutAt">2010-01-01T00:00:00</xsl:attribute>
      <parameters name="email"><xsl:value-of select="$rcv/@email"/></parameters>
    </xsl:element>

  </communication>

</xsl:template>
```

- Wie realisiert man das Mapping von Werten?
- Wie stellt man in einer Benachrichtigungsregel mehrere Benachrichtigungen unterschiedlichen Empfängern gleichzeitig zu, bspw. neben der eigentlichen Meldung noch ein Prüfprotokoll?

Die Umsetzung von fachspezifischen FAQs erfolgt als Funktion im P23R-Entwicklerportal. Jeder angemeldete Benutzer kann fachspezifische FAQs ergänzen, löschen und ändern. Die fachspezifischen FAQs werden versioniert, so dass alte Versionen rekonstruierbar sind. Die fachspezifischen FAQs können kategoriert werden, so dass sie leichter auffindbar sind. Die fachspezifischen FAQ sind Teil des Info-Bereichs im P23R-Entwicklerportal. Die fachspezifischen FAQs ergänzen die allgemeinen FAQs, die durch die Redaktion des P23R-Entwicklerportals gepflegt werden.

E.2.4.3. Lösungsmuster

Die Frequently Asked Questions (FAQ) oder ein spezieller Info-Bereich des P23R-Entwicklerportals bilden auch einen Wissensspeicher, um spezielle Lösungsansätze und -ideen an andere weiter zu geben, beispielsweise spezifische P23R-Infrastrukturen, P23R-Szenarien und Prozessketten. Die Lösungsmuster sollten durch eine Redaktion gepflegt werden. Mögliche Beispiele für Lösungsmuster sind:

- Wie kann man Daten aus dem Internet abfragen?

Man integriert einen speziellen Quelldatenkonnektor in der lokalen P23R-Infrastruktur, der die gewünschten Daten in einem dazugehörigen Pilot-Teildatenmodell liefert. Der Quelldatenkonnektor muss die Funktionalität eines normalem Quelldatenkonnektors realisieren, wobei die Daten nicht aus dem lokalen IT-Fachsystem abgerufen werden, sondern das Ergebnis eines Aufrufs ist, beispielsweise per Webservice aus dem Internet. Der spezielle Quelldatenkonnektor kann pro Mandant natürlich auch so konfiguriert werden, dass beispielsweise eine Autorisierung (bspw. per Nutzer / Passwort) möglich ist. Selbstverständlich muss der spezielle Quelldatenkonnektor sicherheitsmäßig entsprechend in einer passenden Sicherheitszone (DMZ) deployed werden, um auf die Daten im Internet zugreifen zu dürfen.

Dieses Lösungsmuster bedarf keiner speziellen Funktionalität in den Benachrichtigungsregeln, sondern nutzt einfach die vorhandenen Möglichkeiten der Quelldatenkonnektoren zur Datenabfrage.

- Wie nutzt man synchrone erhaltene Rückgabewerte, die man in einem Kommunikationskonnektor zurück bekommt?
- Wie kann man Daten und Antworten von einem Benachrichtigungsempfänger an das IT-Fachsystem übergeben?

E.2.4.4. Prozesse für die optimale Regelerstellung

Die Prozesse zur Regelerstellung, Qualitätssicherung und Freigabe von Benachrichtigungsregelgruppen sind bewusst bisher nicht vorgegeben oder standardisiert, da diese durch fachbereichsspezifische Anforderungen oder organisatorische Rahmenbedingungen angepasst werden können müssen. Mittelfristig wird die Öffentliche Leitstelle wahrscheinlich einen Prozess vorschlagen, der für Normgeber einfach und für viele akzeptabel ist. Darüber hinaus wird es durch Erfahrungen mit verschiedenen Prozessen zur Regelerstellung, Qualitätssicherung und Freigabe aber auch Best-Practices geben, die den Fachdomänen je nach ihren Vorstellungen und Rahmenbedingungen helfen optimierte Prozesse zur Regelerstellung, Qualitätssicherung und Freigabe selbst zu definieren.
Die Veröffentlichung der verschiedenen Prozesse der Fachdomänen und die Dokumentation von Best-Practices in diesem Umfeld sollten anderen bei der Definition aber auch Weiterentwicklung ihrer Prozesse dann helfen.

E.2.5. Fachspezifische Artefakte

Fachspezifische Artefakte stellen domänen-spezifische Beispiele, Dokumentation und Vorlagen dar, beispielsweise für die Domäne Umwelt. Dies bedeutet, dass es sich auch lohnt die generischen, domänen-unabhängigen Artefakte deutlich an die Bedürfnisse der Domäne anzupassen bzw. zu erweitern, da es Fachspezifika gibt.

E.2.5.1. Fachspezifische Musterregel

Die Öffentliche P23R-Leitstelle stellt ein generisches Beispiel für eine Benachrichtigungsregelgruppe in Form der P23R Musterregel bereit. Dieses berücksichtigt aber nicht die Spezifika eines Fachgebietes, bspw. die spezifischen Methoden zur Zuständigkeitsfindung. Daher man eine fachspezifische Musterregel entwickeln, die typische Skript-Dateien, Datenmodelle und Code-Fragmente bereits enthält, so dass bei der Regelerstellung für das jeweilige Fachgebiet diese Musterregel sowohl als Beispiel dienen kann, aber auch als Vorlage, die man den eigenen Bedürfnissen anpassen kann.
Die in P23R4Flex erstellten und analysierten Benachrichtigungsregeln ergaben, dass die Erstellung einer fachspezifischen Musterregel wahrscheinlich keinen Nutzen bietet, da die meisten Probleme sehr allgemein sind und auf alle Benachrichtigungsregeln und -regelgruppen zutreffen.

E.2.5.2. Fachspezifische Handbücher

Neben den Spezifikationen der Technischen Benachrichtigungsregelsprache (T-BRS), den Naming and Design Rules (NDR) einer Leitstelle und dem Handbuch P23R-Regeln für Einsteiger wird es weitere Nachfragen nach erläuternden Unterlagen zur Regelerstellung und Erstellung von Pilot-Teildatenmodellen geben. Die Idee wäre basierend auf den Erfahrungen und Best Practices bei der Regelerstellung in einem Fachgebiet spezielle Handbücher zur Regelerstellung und Erstellung von Pilot-Teildatenmodellen im Fachgebiet zu erstellen, die es Anfängern und Experten des Fachgebietes erleichtern zu erlernen, wie man Benachrichtigungsregeln in diesem Umfeld effektiv erstellt oder verbessern kann. Insbesondere sollten die Handbücher auch spezifische Checklisten mit Aufgaben und Aktivitäten beinhalten, die den Regelersteller und Datenmodellierer helfen, die notwendigen Artefakte in der empfohlenen Reihenfolge zu erstellen und keine Artefakte oder eine der vorgeschlagenen Maßnahmen zu vergessen.
Solche Handbücher machen aber nur dann Sinn, wenn der Erstellungsprozess fachspezifische Aspekte beinhaltet. Die Analyse des bisherigen Vorgehens im Rahmen von P23R4Flex aber auch anderen Piloten hat gezeigt, das zumindest bisher der Erstellungsprozess kaum fachspezifische Aspekte beinhaltet. Daher ist erst einmal das wesentlicher Ziel, die allgemeinen Handbücher und Informationen im P23R-Entwicklerportal zu verbessern.

E.2.5.3. Fachspezifische Vorlagen

Die Vorlagen für Benachrichtigungsregelgruppen, Pivot-Teildatenmodelle und Testsätze erleichtern den Regelerstellern und Datenmodellierern die initiale Erstellung

der selbigen. Die Vorlagen erzeugen die notwendigen Verzeichnisstrukturen, legen die benötigten Dateien an, und enthalten Code-Fragmente, die typischerweise benötigt werden. Neben rein generischer Vorlagen für alle Fachgebiete sollte es auch möglich sein, dass Regelersteller und Datenmodellierer fachspezifische Vorlagen öffentlich bereitstellen können. Diese Vorlagen sollten dann um fachspezifische Code-Fragmente und Vorgaben entsprechend erweitert werden. Insbesondere können die Code-Fragmente auch Platzhalter für Code-Snippets enthalten, die aus den Best-Practices übernommen werden können, insbesondere auch fachspezifische Code-Snippets.
Die Analyse der bisherigen Benachrichtigungsregeln haben gezeigt, das fachspezifische Vorlagen wahrscheinlich wenig Sinn machen, da der überlappende Code wenig fachspezifisch ist. Sehr wohl sollten die vorhandenen Vorlagen überarbeitet werden, um die bisherigen Erfahrungen und gemeinsamen Code-Fragmente in die Vorlagen zu übernehmen. Dabei sollte auch geklärt werden, ob zumindest Platzhalter für Code-Snippets eingefügt werden sollten, um nützliche Code-Snippets der Best-Practices leichter identifizieren zu können.

E.2.6. Code-Dokumentation

Dieser Abschnitt beschreibt, wie durch eine verbesserte Dokumentation von Benachrichtigungsregelgruppen und Pilot-Teildatenmodellen die Regelerstellung bzw. Datenmodellierung und deren Wartung verbessert werden können.

E.2.6.1. Literate Programming

Bisher werden Dokumentationen wie die Anforderungsanalyse, Konzepte und Implementierungsbeschreibungen deutlich vom Programmcode und seinen API-Beschreibungen in zwei Welten getrennt. Wenn man der Idee des ausführbaren Gesetzes näher kommen will, darf man Dokumentation und Code eigentlich nicht trennen. Da der Code bisher nicht durch natürliche Sprache formulierter ist, sollte man wenigstens anfangen diese künstliche Trennung aufzuheben, beispielsweise um dokumentierte Sachverhalte und den dazugehörigen Code an einer Stelle zusammenzufassen und dadurch leichter verstehen und überprüfen zu können.
Diesen Ansatz verfolgt das Konzept des Literate Programming von Donald E. Knuth, das in Kombination mit der Markup Language Markdown mit Github Flavour immer mehr praxistauglich wird. Gute Beispiele bieten die Programmiersprache Coffeescript oder die Datei Readme.md in Projekten bei Github. Hier wird die Dokumentation mit Code gemischt, indem es zwei Ebenen gibt. Der äußere Text in Markdown bildet die Dokumentation, während der Code - durch einem Tabulator eine Ebene eingerückt - in den Text eingebettet werden kann.
Literate Programming ist programmiersprachen-unabhängig und ermöglicht es einen Sachverhalt zu dokumentieren und den dazugehörigen Code direkt einzubetten, so wie es auch in diesem Dokument zu sehen ist. So kann man einerseits ein lesbares Dokument erstellen, sowie gleichzeitig den Code extrahieren und in die Benachrichtigungsregel maschinell ausführbar übernehmen.
Dies ist zunächst eine visionäre Idee, deren Praxistauglichkeit an Beispielen konkret zu überprüfen ist, beispielsweise auch in Kombination mit besser lesbaren Code wie P23R Script ihn bietet.

E.2.6.2. Glossar für die Bezeichner der Datenfelder

Mit Hilfe der im P23R-Entwicklerportal vorhandenen Pivot-Teildatenmodelle wird automatisch ein Glossar (Data Dictionary) erzeugt und regelmäßig aktualisiert. Dazu werden alle Bezeichner, die in den Pivot-Teildatenmodellen verwendet werden, aus den Datenmodellen zusammen mit den zugehörigen Kommentaren extrahiert. Jeder Bezeichner eines Datenfelds oder Datentyps wird im Glossar aufgeführt. Jeder Glossareintrag enthält neben dem Bezeichner als Namen alle Kommentare und die Referenzen, in welchen Datenmodellen der Bezeichner verwendet wird. Zusätzlich kann jeder Glossareintrag noch um eine Kurzbeschreibung und ein zusammenfassende Beschreibung ergänzt werden.
Das Ziel ist es, dass die Bezeichner langfristig mit einer einheitlichen Semantik in den Datenmodellen verwendet werden, und dass man sehen kann in welchem Kontext und in welcher Form diese in vorhandenen Datenmodellen verwendet werden.

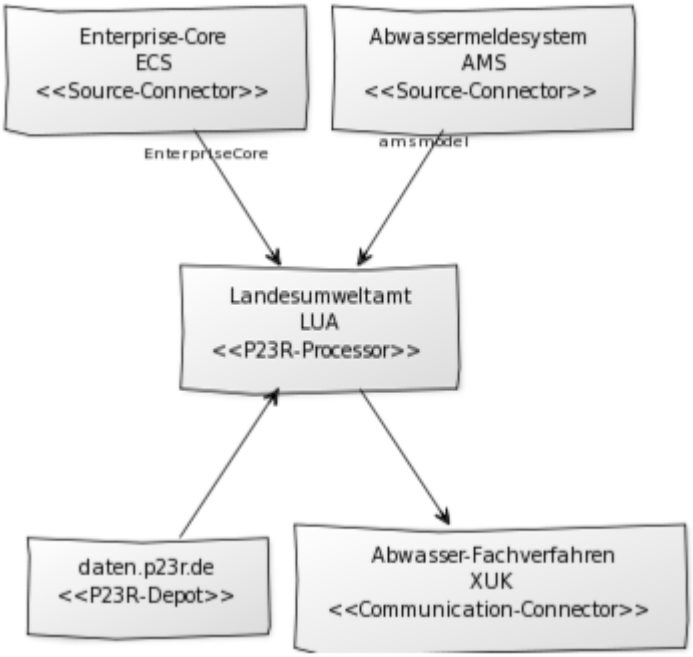
E.2.6.3. Satzpattern

Formale Programmiersprachen sind für Entwickler effektiv zu nutzen aber für Nichttechniker schwer verständlich. Die Nutzung natürlich-sprachlicher Sätze vereinfacht vor allem den Fachexperten zu verstehen, was gemeint ist. Entsprechend ist es bei der Entwicklung von fachspezifischen Benachrichtigungsregelsprachen (F-BRS) ein Ziel, dass sich die Formulierung einer Benachrichtigungsregel der natürlichen Sprache annähert, damit ein Fachexperte den Inhalt einer Benachrichtigungsregel oder eines Pilot-Teildatenmodells leichter lesen und verstehen kann.
Um diesen Ziel näher zu kommen, ist die Verwendung einer "kontrollierten Sprache" hilfreich, die gegenüber der natürlichen Sprache den nutzbaren Wortschatz und die Satzkonstruktionen stark einschränkt. Dadurch entsteht eine endliche Menge an zulässigen Sätzen, deren Erstellung zwar bei Bedarf auch technisches Know-How voraussetzt, die dann aber maschinell in die T-BRS übersetzt werden können soll. Da die kontrollierte Sprache natürlich-sprachliche Sätze verwendet, sollten diese dann auch von Nichttechnikern problemlos verstanden werden können, wenn die gleiche Semantik zugrunde liegt.
Ein möglicher Lösungsansatz ist die Definition einer Menge von zulässigen Sätzen in Form von Satzpattern. Ein Satzpattern ist ein korrekter natürlich-sprachiger Satz mit Platzhaltern. Jeder Satz kann genau einer Funktion zugeordnet werden, beispielsweise einer Funktion, die in P23R Script implementiert wurde. Die Platzhalter können eineindeutig auf die Argumente der zugehörigen Funktion abgebildet werden. Somit entspricht die Erstellung eines Satzes in einem Satzpattern technisch dem Aufruf einer Funktion in P23R Script.
Das Vorgehen ist es erst einmal die technisch benötigten Funktionen in P23R Script durch Techniker erstellen zu lassen. Die Fachexperten können nun (mit ein wenig Hilfe der Techniker) die fachlichen Aussagen in natürlicher Sprache erstellen, wobei nur vorab definierte Satzmuster der kontrollierten Sprache zulässig sind. Ein Compiler kann am Ende die fachlichen Sätzen in normale Funktionsaufrufe umsetzen und letztlich die benötigten Skripte in XSLT und P23R Selection erzeugen, die in die Benachrichtigungsregel eingebunden werden.
Damit gibt es eine Aufgabenteilung zwischen den Technikern und den Fachexperten. Die Fachexperten schreiben in verständlichen Sätze die wesentlichen Fakten mit

Hilfe der Satzmuster. Die Techniker stellen die benötigte Basisfunktionen mit den dazugehörigen Satzmustern bereit. Dies sollte dann insgesamt zu leichter lesbaren Benachrichtigungsregeln führen, ohne die technischen Möglichkeiten einzuschränken.
Zur Demonstration des Ansatzes folgt ein Beispiel für die formale Beschreibung einer P23R-Infrastruktr, wie sie für die Ausführung einer spezifischen Benachrichtigungsregel benötigt und entsprechend voraus gesetzt wird.

```
   [p23rInfrastructureDescription]

# Infrastruktur

Das Diagramm setzt die Benachrichtigungsregelgruppe '...' kurz XUK für die Umsetzung der Norm ... um.

  Der P23R Prozessor 'Landesumwelt' kurz LUA nutzt Daten vom
  Datenquellkonnektor 'Abwassermeldesystem' kurz AMS mit dem Datenmodell amsmodel und
  Datenquellkonnektor 'Enterprise-Core' kurz ECS mit dem Datenmodell EnterpriseCore,
  und kommuniziert mit dem
  Kommunikationskonnektor 'Abwasser-Fachverfahren' kurz XUK,
  sowie nutzt er den Zuständigkeitsfinder daten.p23r.de .
```

Dieser Text kann automatisch in ein UML-Diagramm übersetzt werden.



Die Grafik kann automatisch erzeugt werden, in dem mit Hilfe der Satzpattern die Grafik schrittweise beschrieben wird. Dies wird durch die nachfolgenden Beispiele exemplarisch dargestellt, die eine Teilmenge der benötigten Satzteile mit der Anbindung an die technischen Funktionen definiert.

```
'Der P23R Prozessor <<string:processorName>> kurz <<identifier:processorId>> nutzt Daten vom'
p23rInfrastructure.setProcessor(processorName, processorId) -> ...
```

```
'Datenquellkonnektor <<string:sourceConnectorName>> kurz <<identifier:sourceConnectorId>>
   mit dem Datenmodell <<identifier:model>> und'
p23rInfrastructure.setSourceConnector(sourceConnectorName, sourceConnectorId, model) -> ...
```

Die technischen Funktionen könnten dann beispielsweise den folgenden YUML-Code erzeugen, der durch den Dienst in die obige Grafik übersetzt wird.

```
[Abwassermeldesystem\nAMS\n<<Source-Connector>>]amsmodel->
  [Landesumweltamt\nLUA\n<<P23R-Processor>>]
[Enterprise-Core\nECS\n<<Source-Connector>>]EnterpriseCore->
  [Landesumweltamt\nLUA\n<<P23R-Processor>>]
[Landesumweltamt\nLUA\n<<P23R-Processor>>]->
  [Abwasser-Fachverfahren\nXUK\n<<Communication-Connector>>]
[Landesumweltamt\nLUA\n<<P23R-Processor>>]<-
  [daten.p23r.de\n<<P23R-Depot>>]
```

Die Umsetzung von Satzpattern ist weniger ein technisches Problem. Vielmehr muss deren Praxistauglichkeit und Nutzen bei Verwendung in Benachrichtigungsregeln und Pilot-Teildatenmodellen durch konkrete Beispiele überprüft werden. Natürlichsprachliche Satzpattern könnten aber deutlich dabei helfen, die Lücke zwischen einer rein technischen Beschreibung und den Fachexperten zu verringern.

E.2.7. Automatischer Regelerstellungsassistent

Für neue Regelersteller ist es wichtig, Unterstützung bei der Erstellung ihrer ersten Benachrichtigungsregeln, Pivot-Teildatenmodelle und Testsätze zu erhalten. Im Idealfall werden sie Schritt-für-Schritt mit Hinweisen durch die Erstellung einer Benachrichtigungsregel oder einem Pivot-Teildatenmodell geleitet. Analog zu Fahrerassistenzsystemen im Auto, bei denen der Fahrer nicht bevormundet, aber im Bedarfsfall unterstützt wird, sollte das P23R-Entwicklerportal die Erstellung assistieren.
Dazu sollte das P23R-Entwicklerportal eine Benachrichtigungsregel, Pivot-Teildatenmodell oder Testsatz analysieren um den Stand, d.h. die Qualität derselbigen, zunächst einmal zu ermitteln. Daraus resultieren letztlich Qualitätsmerkmale, die erfüllt sind oder nicht. Qualitätsmerkmale können verpflichtend, empfehlenswert oder optional sein. Auf Basis der Qualitätsmerkmale würde dann dynamisch eine Checkliste erstellt werden, die die nächsten Schritte entsprechend ihrer Priorität ermittelt. Technisch kann dies beispielsweise durch eine allumfassende, globale Checkliste erfolgen, wobei mit jedem Eintrag, d. h. einer Aktivität, konkret umzusetzende Qualitätsmerkmale verknüpft sind. Die Aktivitäten haben außerdem Prioritäten, damit die nächsten Schritte identifiziert werden können.
Der automatische Regelerstellungsassistent prüft regelmäßig die Qualitätsmerkmale und aktualisiert die noch durchzuführenden Aktivitäten auf der Checkliste. Die resultierende Checkliste hilft dem Regelersteller welche Aktivitäten vorrangig erfolgen sollten, um die Qualität der Benachrichtigungsregel, des Datenmodells oder Testsatzes zu erhöhen und dadurch zu einem zu veröffentlichenden Artefakt zu kommen. Eine Veröffentlichung kommt dann in Frage, wenn alle Qualitätsmerkmale erfüllt wurden, die mit verpflichtend gekennzeichnet sind.

Technisch ist solch ein automatischer Regelerstellungsassistent umsetzbar. Der Aufwand liegt in der Implementierung der Messfunktionen zur Ermittlung der Qualitätsmerkmale. Diese werden auch generell für die Qualitätssicherung von Benachrichtigungsregelgruppen, Pilot-Teildatenmodelle und Testfälle benötigt. Um aber frühzeitig zu nutzbaren Lösungen zu kommen, sind in einem ersten Schritt umfassende Checklisten für die obigen Artefakte zu erstellen, die alle notwendigen Aktivitäten aufführen, priorisieren und die charakterisierenden Qualitätsmerkmale auflisten.

E.2.8. Fazit

Die Analyse der bisher erstellten Benachrichtigungsregeln insbesondere aus der Domäne Umwelt hat zwar ergeben, dass die Erstellung fachspezifischer Funktionsbibliotheken für die Domäne keine wesentlichen Erleichterungen bei der Regelerstellung bringen dürfte, es aber diverse andere Maßnahmen gibt, die man umsetzen sollte. Aus der Analyse der bisherigen Arbeiten sollten für die Domäne Umwelt vor allem allgemeine Maßnahmen realisiert werden, die möglichst auch kurzfristig umsetzbar sind. Dies betrifft vor allem die Dokumentation und Vorlagen. Ein Schwerpunkt bei der Erstellung von Benachrichtigungsregeln der Domäne sind zurzeit Testen und Validierungen. Mit dem im P23R-Entwicklerportal integrierten Test-P23R-Prozessor und den verbesserten Testfunktionalitäten sind bereits im laufenden Prozess die Erstellung von Benachrichtigungsregeln ohne eine eigene P23R-Lösung und die Qualitätssicherung von Benachrichtigungsregeln deutlich verbessert worden. Somit sollte die Vereinfachung des Schreibens von Validierungen in der Domäne Umwelt ein nächster Schritt sein.
Die Erstellung fachspezifischer Artefakte, wie Musterregeln, Handbücher und Vorlagen wird entsprechend der Analysen als wenig hilfreich gesehen. Allerdings die Verbesserung der allgemeinen Handbücher und Vorlagen sind klare Verbesserungen, die auch kurzfristig umgesetzt werden sollten. Da die Realisierung eines automatischen Regelerstellungsassistenten kurzfristig nicht machbar ist, sollte aber zumindest die Dokumentation von Checklisten zur Erstellung von Benachrichtigungsregelgruppen, Pilot-Teildatenmodellen und Testfällen zeitnah voran getrieben werden.
Ein weiterer Schwerpunkt ist die Unterstützung der Dokumentation von Best-Practices im P23R-Entwicklerportal, um vor allem fachspezifische FAQs, Lösungspattern und Code-Snippets zu sammeln.
Die fachspezifischen Benachrichtigungsregelsprachen sind mittelfristig ebenfalls eine vielversprechende Option, da auch sie die Arbeit der Regelersteller und vor allem Reviewer von P23R-Artefakten wesentlich erleichtern sollten. Für die Datenmodellierer steht mit P23R Model bereits eine erste Lösung im P23R-Entwicklerportal bereit, die im Sinne des Literate Programming leicht in die Gesamtdokumentation eines Pivot-Teildatenmodells eingebettet werden kann. Aus Sicht der Domäne Umwelt aber sehr wohl auch aus anderen Domänen sollte vor allem eine fachspezifische Sprache, die funktionell auf der Idee der Object Constraint Language (OCL) sowie einer homogenisierten Syntax wie P23R Script aufbaut, priorisiert werden, um den Aufwand zum Schreiben von Validierungen zu vereinfachen.
Zusammenfassend kann die Erstellung der P23R-Artefakte in einem ersten Schritt durch die Optimierung der vorhandenen Dokumentation, der Erstellung von Checklisten und der Erweiterung der Funktionalitäten des P23R-Entwicklerportals unter-

stützt werden. Mittelfristig ist die Entwicklung von fachspezifischen Sprachen ein nächster Schritt. Eine Automatisierung der Qualitätssicherung und daraus abgeleitet ein automatischer Regelerstellungsassistent sind langfristig ebenfalls machbar, selbst wenn der Wunsch danach eine hohe Priorität besitzt.

E.3. Analyse von XML-Schemasprachen für die Qualitätssicherung im Kontext des P23R-Prinzips

Das P23R-Prinzip setzt auf XML-Validierung nach XML Schema Version 1.0[1]. Es gibt weitere Mechanismen der XML Dokumentprüfung, die hier[2] darauf hin untersucht werden, ob sie den Nutzen des P23R-Prinzips durch verbesserte Prüfungen steigern können. Daneben ist auch beachtet worden, ob freie Implementierungen (mit offenen Lizenzen) dieser Mechanismen vorliegen.
Es werden die XML-Schemasprachen Schema 1.1, Relax NG, Schematron und DSD darauf hin untersucht, ob ihre Verwendung im P23R erwogen und die Spezifikation des P23R dementsprechend erweitert werden sollte[3].

E.3.1. Vorgehensweise

Zunächst werden Kriterien festgelegt, die eine strukturierte und somit vergleichbare Bewertung von XML-Schemasprachen im Kontext des P23R-Prinzips ermöglichen. Dann werden die o.g. Kandidaten bezüglich der festgelegten Kriterien bewertet und dokumentiert. Die dabei gewonnenen Erkenntnisse sowie ggf. besonders zu würdigende Merkmale werden in einer Zusammenfassung aufbereitet und verdichtet.

E.3.2. Bewertungskriterien

Die für die vergleichende Bewertung von XML-Schemasprachen zu verwendenden Kriterien werden in einem iterativen Bewertungsprozess unter Berücksichtigung der DIN 66272 (Qualitätsmerkmale für Software, speziell nicht funktionale Anforderungen) ausgewählt. Ausgangsbasis ist folgende Liste nicht funktionaler Anforderungen:

- Zuverlässigkeit (Systemreife, Wiederherstellbarkeit, Fehlertoleranz)
- Benutzbarkeit (Verständlichkeit, Erlernbarkeit, Bedienbarkeit)
- Leistung und Effizienz (Antwortzeiten, Ressourcenbedarf, Wirtschaftlichkeit)
- Wartbarkeit, Änderbarkeit (Analysierbarkeit, Stabilität, Prüfbarkeit, Erweiterbarkeit)
- Flexibilität (Unterstützung von Standards)

[1] Vgl. http://www.w3.org/TR/xmlschema-1/ und XML Schema Versionsübersicht unter http://www.w3.org/standards/techs/xmlschema#w3c_all

[2] im Rahmen des Arbeitspakets 9.3 („Analyse der implementierten Prozesskette...“, Unterpunkt „... Kontext QS im Feld der Umweltberichterstattung“)

[3] Die Zahl der XML-Schemasprachen ist überschaubar, weshalb eine elaborierte und kriteriengestützte Auswahl nicht notwendig ist.

- Skalierbarkeit (Änderungen des Problemumfangs bewältigen)

Zur Vermeidung des Vendor Lock-In wird das Vorhandensein einer freien Implementation positiv bewertet.

E.3.3. XML-Schemasprachen

E.3.3.1. Schema 1.1

XML Schema ist eine vom W3C empfohlene Schemasprache zur Definition und Validierung von XML Dokumenten, dessen Entwicklung bis in das Jahr 2001 (XML Schema: Formal Description) zurückreicht. Die hier betrachtete Version 1.1 ist im Jahr 2012 erschienen und ergänzt die Implementierung um weitere, aus anderen Schemasprachen bekannte, Features. Aufgrund der langen Verbreitung hat sich XML Schema als eine bewährte Methode zur Definition und Validierung etabliert.
Durch die große Anzahl von bereits definierten Datentypen lassen sich entsprechende Inhalte leicht validieren. Es ist ebenso möglich, eigene, fachspezifische Datentypen zu definieren, diese mit Hilfe von include-Anweisungen zu verschachteln und so einen XMLBaukasten zu erstellen. Diese Möglichkeit wurde z.B. intensiv bei der Erstellung der XÖV Standards XUBetrieb und XUKommunalabwasser genutzt. Weiterhin ist durch die Verwendung von regulären Ausdrücken, Kardinalitäten sowie kontextabhängiger Definitionen eine weitreichende Validierung von XML Dokumenten möglich.
Der Anspruch, möglichst alle auftretenden Probleme mit einer Schemasprache abzudecken, führt dazu, dass XML Schema an einigen Stellen als ein Kompromiss aus vielen Vorschlägen anzusehen ist. Die dadurch entstehende Komplexität trägt nicht zur Benutzerfreundlichkeit und einfachen Erlernbarkeit bei.
Da XML Schemata in Form eines XML-Dokuments beschrieben werden, ist eine elektronische Verarbeitung mit gängigen Bibliotheken möglich.

E.3.3.2. Relax NG

RELAX NG (Regular Language Description for XML New Generation) ist eine weitere XMLSchemasprache, welche ebenso wie XML Schema Strukturen in XML-Syntax definiert.
Darüber hinaus ist jedoch auch die Verwendung einer eigenen, semantisch äquivalenten, Sprache möglich, die sich durch eine kompaktere Syntax auszeichnet. Im Gegensatz zu XML SCHEMA kennt RELAX NG nur zwei Datentypen (text und token). Da RELAG NG nur Aussagen darüber trifft, ob das vorliegende Dokument konform zum verwendeten Schema ist, sind bewusst keine Vorgabewerte wie in XML Schema möglich. RELAX NG legt den Fokus auf die Strukturdefinition von XML-Dokumenten und ist somit nicht zur alleinigen Verwendung in der Dokumentvalidierung geeignet. Hierzu wird nach ISO/IEC 19757 eine komplementäre Lösung aus RELAX NG und SCHEMATRON empfohlen.

E.3.3.3. Schematron

SCHEMATRON unterscheidet sich von den bereits vorgestellten XML-Schemasprachen XML-SCHEMA und RELAX NG durch die Tatsache, dass Schematron einzig zur Vali-

dierung von Inhalten konzipiert ist. Erst in Kombination mit einer XML-Schemasprache wie etwa RELAX NG (komplementäres Validierungsverfahren nach ISO/IEC 19757) lässt sich die komplette Bandbreite abdecken.
Die Anforderungen an eine einfache und effektive Validierung von XML-Dokumenten erfüllt SCHEMATRON jedoch in allen Punkten. Durch die Verwendung von XPATH Ausdrücken bietet Schematron eine Präzision, die allen anderen, auf Grammatiken basierten Lösungen fehlt.
Der Einsatz von Mustern ermöglicht eine einfache Umsetzung von komplexen Regelwerken, welche ebenso einfach erweiterbar sind. SCHEMATRON bietet die Möglichkeit, Validierungsergebnisse als Text oder XML-Dokument auszugeben. Durch die Verwendung von XSLT (Extensible Stylesheet Language Transformations) ist eine flexible Weiternutzung innerhalb bestehender Prozessketten möglich.

E.3.3.4. DSD

Die Document Structure Description (DSD) ist eine XML Schemasprache, deren Hauptentwicklungziel in der einfachen Benutzbarkeit liegt. Durch die Verwendung von wenigen und einfach zu verstehenden Sprachbestandteilen ist es auch Anfängern möglich, DSD Dokumente zu verstehen bzw. selbst zu definieren.
Die Beschreibung von Regeln erfolgt mit Hilfe von boolescher Logik und regulären Ausdrücken.
Aufgrund der Ausrichtung auf schnelle Erlernbarkeit und der Verwendung einfacher Strukturen, stößt DSD in größeren Umgebungen schnell an seine Grenzen.

E.3.4. Zusammenfassung

Die hier vorgestellten Schemasprachen verfolgen teilweise unterschiedliche Ansätze und sind oft nur in Kombination untereinander vergleichbar. So lässt sich der Funktionsumfang von XML SCHEMA 1.1 am besten mit einer Lösung aus RELAX NG und SCHEMATRON vergleichen. Durch den Fokus auf bestimmte Teilaspekte ist die komplementäre Lösung aus RELAX NG und SCHEMATRON in Sachen Funktionsumfang und Benutzbarkeit XML SCHEMA überlegen.
Zwar ist XML SCHEMA eine über viele Jahre etablierte Technologie zur Validierung von XML Dokumenten, zeigt jedoch auf Grund seines großen Funktionsumfangs deutliche Defizite in Wartbarkeit und Erweiterbarkeit. Die Trennung von Strukturdefinition und Inhaltsprüfung (RELAX NG & SCHEMATRON) führt zur einer besseren Übersicht und Skalierbarkeit. Durch die Verwendung von XPATH-Ausdrücken lassen sich präzise Prüfungen realisieren.
Die aus den vorangegangenen Punkten gewonnene Flexibilität und Skalierbarkeit macht eine Lösung aus RELAX NG und SCHEMATRON zu einem idealen Kandidaten für den P23R. Der Einsatz von DSD scheidet aufgrund des ähnlichen Funktionsumfangs wie XML SCHEMA aus. DSD bietet zwar unter Verwendung von regulären Ausdrücken die Möglichkeit, komplexere Regelwerke zu definieren, kommt jedoch nicht an den Funktionsumfang von SCHEMATRON mit XPATH Ausdrücken heran.

E.4. Testweise Regelerstellung für weitere Berichtspflichten

Das folgende Kapitel bezieht sich auf die Arbeiten des Auftragnehmers ENDA.
Ziel war es, unter Nutzung der zuvor definierten Regelkomponenten Regeln für die Abwicklung so vieler Berichtspflichten bzw. Berichtsprozessteile wie irgend möglich zu erstellen.
Nun konnten allerdings während der Projektlaufzeit keine solchen Regelkomponenten identifiziert werden[4].Auch stellten sich die Arbeiten an der zu verwendenden Musterimplementierung als umfangreicher und zeitaufwändiger heraus als ursprünglich kalkuliert. Entsprechend der Einschätzung im Angebot konnte daher nur eine Berichtspflicht umgesetzt werden. Um eine Berichtspflicht auszuwählen, die gleichzeitig relevant, fachlich aber von im Projektvolumen darstellbarem Umfang ist, wurde beschlossen, eine der TOP Berichtspflichten – die 13. BImSchV – exemplarisch herauszugreifen, anstatt (wie ursprünglich geplant) mittels eines Scoring-Systems zu verfahren. Auf diese Weise liegt nun eine beispielhafte Umsetzung einer Berichtspflicht vor, und die benötigten Werkzeuge und Abläufe sind auf eine Weise dokumentiert, die die Umsetzung weiterer Berichtspflichten deutlich vereinfacht.

E.4.1. Vorgehensweise

Um die Erstellung von Regeln zu vereinfachen, wurde seitens Fraunhofer FOKUS das P23R-Entwicklerportal (zu erreichen unter https://entwickler.p23r.de) und ein dort hinterlegter Leitfaden „P23R-Regeln für Einsteiger“ bereit gestellt. Das Entwicklerportal ermöglicht es, zuvor selbst entwickelte Regeln in ein vom P23R-Prozessor lesbares Format umzuwandeln („paketieren“) und unter einer temporären Internet-Adresse zu veröffentlichen. Das genannte Format ist in der T-BRS, der technischen Benachrichtigungsregelsprache, spezifiziert. Es ist allerdings im Fluss, und die aktuelle Musterimplementierung verlangt an einigen Stellen ein anderes Format als in der veröffentlichen T-BRS angegeben. Eine angekündigte Version 1.1 ist bislang nicht erschienen; die 1.0 siehe unter http://leitstelle.p23r.de/documents/2013-01-09_P23R_Spezifikation_T-BRS.pdf. Weiterhin kann man fertige Regeln mit Testdaten validieren. Der Leitfaden erläutert die Bedienung des Entwicklerportals und stellt die Dateien einer einfachen Beispielregel vor.
Die Erstellung einer Regel war mittels des Leitfadens und zusätzlicher Workshops erfolgreich. Zugleich wurden viele offene Fragen diskutiert.
Aus dem Projekt wird die Klärung der Lizenzfrage für das Entwicklerportal gefordert. Die Musterimplementierung sollte unter eine freie Lizenz gestellt werden. Damit ist ein Hosting des Entwicklerportals momentan und ggf. auch in Zukunft nur durch Fraunhofer FOKUS möglich. Um auch unter ungünstigen Umständen eine weitere Nutzbarkeit der Projektergebnisse zu erreichen, wurde daher mit den erhaltenen Informationen und der T-BRSDokumentation die Paketierungsfunktion reimplementiert. Sie liegt in Form eines Perl- Skripts vor, das im öffentlichen ENDA-Repository herunter geladen werden kann (`http://devel.enda.eu/websvn/filedetails.php?repname=ENDA+%C3%B6ffentliches+Subversion-Repository&path=%2Fsvnroot%2Fp23r%2Fp23r4flex%2Ftrunk%2Fregel%2Fpaketierer`).

[4]Die Entwicklung von Regelkomponenten lag nicht im Aufgabenbereich von ENDA.

Der reimplementierte Paketierer arbeitet auf einer einfachen Verzeichnisstrukur; notwendige Informationen, die im Entwicklerportal in Form von Manifest.xml-Dateien angegeben werden müssen, werden im Baum abgelegten „info.txt“-Dateien entnommen, die ausführlich kommentiert sind. Sinn und Zweck jedes Regelbestandteils ist hier mit hoher Informationsdichte dokumentiert.
Wird der Paketierer aufgerufen, berechnet er benötigte Prüfsummen automatisch, erstellt ggf. fehlende, aber mit sinnvollen Defaults ersetzbare, Dateien und warnt, wenn erforderliche Bestandteile oder Informationen fehlen. Das Ergebnis wird automatisch, wenn möglich (Hierzu wird ssh-Zugriff auf den Leitstellenserver mit hinterlegtem Schlüssel benötigt), auf die in der obersten info.txt-Datei angegebene Leitstelle hochgeladen. So paketierte Regeln sind zur Musterimplementierung in Version 3.2.0 und 4.0.0 kompatibel, nicht aber zur T-BRS in Version 1.0 (s.o.). Dies stellt aber kein Problem dar, da die T-BRS ohnehin überarbeitet wird, um Inkonsistenzen zu korrigieren. Für das hier beschriebene Projekt stand im Vordergrund, Ergebnisse zu erzielen, die durch existierende Software nutzbar sind.
Um den Paketierer zu testen und sein Eingabeformat praxisnah zu dokumentieren, wurde die in Teil 1 des Projekts durch Jeroen Gremmen von der Firma ATOS entwickelte Regel „XUKommunalabwasser-Berichterstattung“ erfolgreich aus den Entwicklerportal-GITRepositories in eine Paketierer-Verzeichnisstruktur überführt und getestet.
Um die Bestandteile der Regel für die 13. BImSchV – v.a. Datenmodelle und Transformationen – zu erstellen, wurde auf die sehr nützlichen Ergebnisse aus der ersten Pilotierung des P23R zurück gegriffen (Siehe unter http://mlf.p23r.de/modul-6-fachliche-konzeption-in-arbeit/#c308 „Anwendungsbeispiel Datenmodell XUEmission“).
Der Ablauf der Regel ist wie folgt:

- Eine eingehende Nachricht wird verarbeitet; diese enthält Bundesland und Berichtsjahr. Ihr Namespace ist http://enda.eu/namespace/p23r/13.BImSchV/Message.
- Das Datenselektionsskript Selection.xquery läuft ab. Es fordert Daten vom Quellkonnektor für den Namespace http://www.xubetrieb.de/schema/xugfa_kompakt/output_sc/0_1_0 an.
- Der Quellkonnektor liest aus der XML-Datenbank auf dem P23R-Prozessor ein hinterlegtes XUEmission-Dokument mit den Daten des betreffenden Jahres und für das gewählte Bundesland. Dieses Dokument wird noch im Quellkonnektor in ein fachliches Zwischenformat, „XUGFA kompakt“, konvertiert. Das Ergebnis liegt im Namespace http://www.xubetrieb.de/schema/xugfa_kompakt/output_sc/0_1_0, der vom Selektionsskript angefordert wurde.
- Das Datenselektionsskript legt die Ergebnisse des Quellkonnektors in einem <selected>-Element ab; das Ergebnis hat den Namespace http://enda.eu/namespace/p23r/13.BImSchV/Step0.
- Der Namespace des Ergebnisdokuments ist als Eingabeformat für den Transformationsschritt Nr. 1, „Umwandlungsschritt“, hinterlegt. Durch sukzessives Testen, welchem Schema das aktuelle Dokument genügt, ermittelt die P23R-Regelmaschine den auszuführenden Schritt. Dieser läuft so ab: Zunächst wird aus dem aktuellen Datenprofil (Das Profil enthält Metadaten, insbesondere die

Informationen aus der auslösenden Nachricht. Wenn man die transformierten Daten mit einem Brief vergleicht, entspricht das Profil in etwa den Informationen auf dem Umschlag, das Inhaltsdokument dem innen liegenden Brief.) und dem aktuellen Inhaltsdokument ein gemeinsames XML-Dokument erzeugt. Dieses wird mit zwei verschiedenen XSLT-Skripten transformiert – das erste XSLT erzeugt das neue Profil, das zweite das neue Inhaltsdokument. Der Umwandlungsschritt fügt dem Datenprofil eine Zieladresse für den Import und die benötigten Login-Daten hinzu. Das Inhaltsdokument wird von XUGFA kompakt in das BUBE XML1-Format umgewandelt und in ein <notification>-Element gekapselt. Der Namespace dieses neuen Inhaltsdokuments, http://enda.eu/namespace/p23r/13.BImSchV/Notification, wird dann wieder mit den Eingabeformaten aller definierten Transformationsschritte verglichen[5] .

- Das erzeugte Dokument entspricht keinem der Eingabeformate von Transformationsschritten, wohl aber dem ebenfalls in der Regel enthaltenen Schema Notification.xsd. Dieser Fall ist speziell und zeigt der P23R-Regelmaschine an, dass die Verarbeitung beendet ist und das aktuelle Inhaltsdokument die fertige Benachrichtigung ist. Die Verarbeitung stoppt.
- Nun kann der Nutzer das Ergebnisdokument in einem P23R-Client sichten und, wenn es seiner Prüfung standhält, zur Weitergabe freigeben.
- Nach Freigabe wird der CommunicationConnector aktiv und importiert das XML1 über einen RESTful Web Service ins BUBE-System.

Aus diesem Ablaufplan ergibt sich, welche Schemata, Skripte, Programme und sonstigen Dokumente für das Erstellen der gewünschten Regel notwendig sind.

XUGFA kompakt-Format

Als nächstes wurde das fachliche Format „XUGFA kompakt“ fürs Berichten der 13. BIm- SchV entwickelt. Um eine schlanke, leicht verständliche Regel zu erhalten, wurden ausschließlich Pflichtfelder in XUGFA kompakt aufgenommen. Das Ergebnis ist in Abb. E.2 dargestellt.

Quellkonnektor

Dann wurde der benötigte Quellkonnektor in der Programmiersprache Perl entwickelt. Um bestehende Ergebnisse zu nutzen, wurde die Grundfunktionalität des XUKommunalabwasser-Quellkonnektors übernommen und ein externes Modul für die Konvertierung in XUGFA kompakt programmiert. Es wird beabsichtigt, weitere Berichtspflichten gleichartig abzudecken und die Spezifika analog zur 13. BImSchV in einzelnen Modulen abzulegen.

Arbeiten an der Musterimplementierung

Um den Verlauf der Verarbeitung besser verfolgen zu können – die P23RMusterimplementierung liefert nur Logs, wenn Schritte erfolgreich ablaufen, ansonsten muss man sich mit

[5] Der Namespace des Profils bleibt während der gesamten Regelverarbeitung konstant http:// leitstelle.p23r.de/NS/p23r/nrl/ NotificationProfile1-0.

Java-Exceptions und der Programmstelle, an der sie aufgetreten sind, begnügen – wurde die Musterimplementierung so erweitert, dass jede

- Validierung
- Transformation
- Abfrage

jeweils die beteiligten XML-Dateien (Schema: „.xsd“, Stylesheet: „.xslt“, XQuery- Dokument: „.xquery“ und das verarbeitete Dokument) in einem temporären Verzeichnis ablegt und ein kurzes Ablaufdokument erstellt wird.
Weiterhin wurde an etlichen Stellen, an denen Fehler auftraten und durch Sichten des Logfiles vom Application Server identifiziert werden mussten, aussagekräftige Fehlermeldung anstelle von Exceptions implementiert.

Regel

Für die Regel wurden Schemata für die auslösende Nachricht (Message.xsd), das Ergebnisformat und das von ihm eingebundene BUBE-Format (Notification.xsd, bube_xml_1.xsd) sowie das vom Quellkonnektor gelieferte XUGFA kompakt (xugfa_kompakt_nachrichten.xsd) erstellt und abgelegt. Weiterhin wird das Selektionsskript Selection.xquery benötigt, das die initialen Daten anfordert. Der P23R selektiert die Daten laut T-BRS 1.0 und auch 1.1 in einer von Fraunhofer FOKUS eigens entworfenen Abfragesprache, P23R Selection, ehemals DataSel genannt. Da für diese Sprache nur eine Syntaxbeschreibung und ein syntaktisches Prüfwerkzeug, aber kein Compiler, Interpreter oder anderes lauffähiges Werkzeug existiert, nutzt die Musterimplementierung nicht P23R Selection, sondern als Übergangslösung bis zur Implementierung von P23R Selection die Standardsprache XQuery.

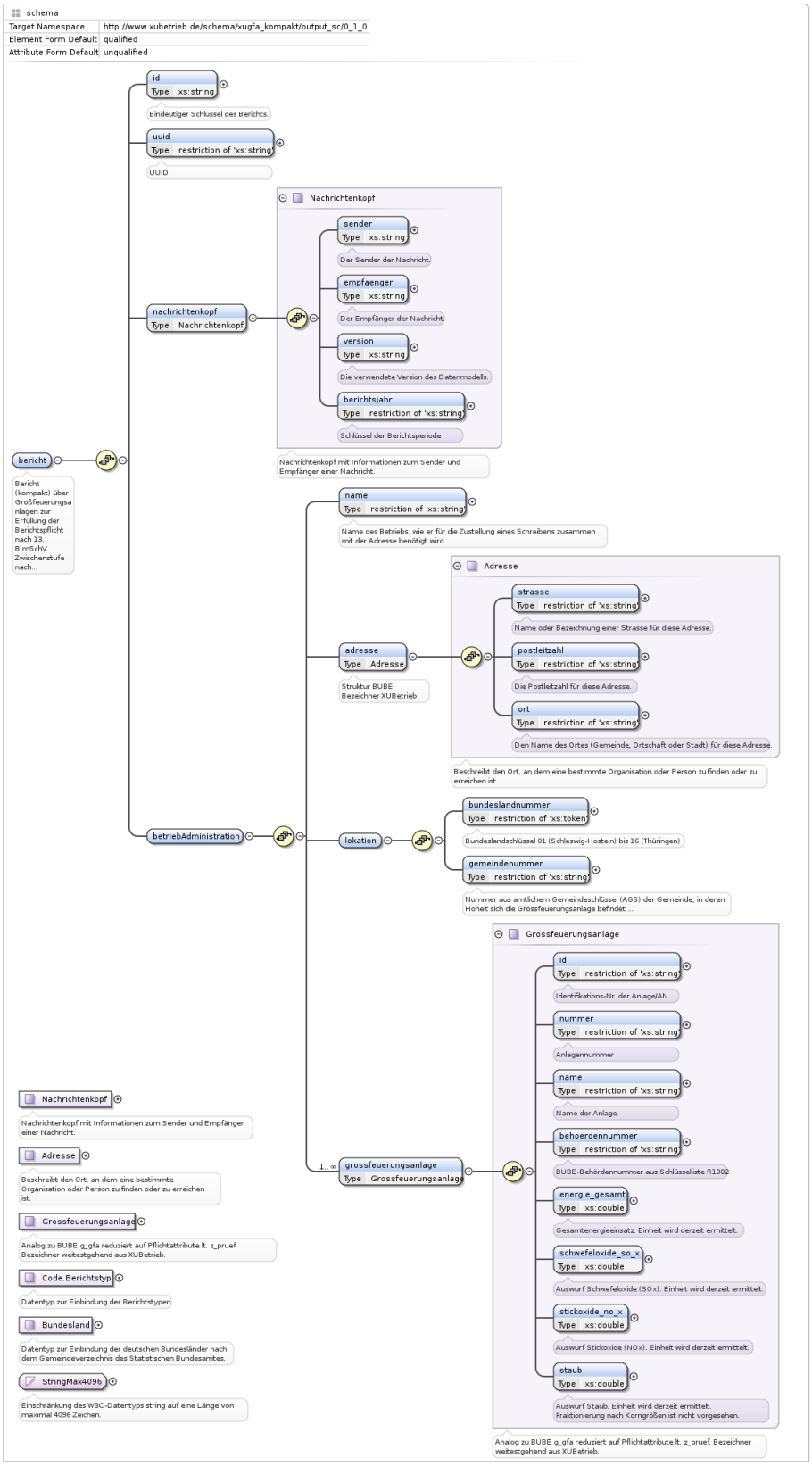


Abbildung E.2.: XUGFA kompakt-Modell für die 13. BImSchV

Schließlich wurden für die Transformation von Profil und Nachricht zwei Transformationsskripte im XSLT-Format erstellt:

1. ProfileTransformationStep0.xslt Im Profil werden durch diese Transformation der zu verwendende CommunicationConnector durch eine ID sowie einige hart codierte Daten hinterlegt – Zieladresse für den Datenimport in BUBE, Benutzername und Kennwort – die der CommunicationConnector nutzt, um die freigegebene Benachrichtigung zuzustellen.

2. NotificationTransformationStep0.xslt Diese Transformation arbeitet auf den Berichtsdaten und wandelt das Format XUGFA kompakt in BUBE XML1 um. Das Ergebnis wurde validiert und entspricht dem aktuellen Stand von XML1, Release 2.1.6 vom 11.11.2013.

CommunicationConnector

Für den CommunicationConnector wurde ebenfalls bestehender Code als Basis genutzt, nämlich der für den XUKommunalabwasser-Piloten erstellte CommunicationConnector. Dieser enthält bereits die Funktionalität, ein Ergebnisdokument an einen RESTful Web Service hochzuladen.
Das REST-Interface, das hierfür auf BUBE[6]-Seite benötigt wird, steht während der Projektlaufzeit noch nicht zur Verfügung. Es ist allerdings genau spezifiziert und wird derzeit (Mai 2014) durch die Firma QuinScape GmbH entwickelt. Es wird daher davon ausgegangen, dass nur minimale Anpassungen am CommunicationConnector nötig sein werden, um einen BUBE-Import zu gewährleisten.

P23R-Client

Ein P23R-Client ist eine Anwendung, die es erlaubt, Regeln zu aktualisieren, zu aktivieren und auszulösen, die erstellten Benachrichtigungen zu sichten und bei Korrektheit freizugeben.
Als solcher ist ein P23R-Client kein Teil der Regelentwicklung, allerdings nützt der P23R-Prozessor auch mit Regeln nicht, wenn kein Client zur Bedienung vorhanden ist. Es existiert ein generischer Client, der Teil der Fraunhofer FOKUS-Musterimplementierung ist. Dieser ist in der vorliegenden Form für Pilotprojekte geeignet, nicht aber für Endanwender.
Um ein Grundgerüst für einen benutzerfreundlichen P23R-Client mit ansprechender Oberfläche zur Verfügung zu haben, wurde im Projekt ein Client mit minimaler Benutzerverwaltung entwickelt, der Theming durch CSS-Style Sheets zulässt und nicht mehr als die oben genannten Funktionalitäten bietet. Unnötige Komplexität, wie die Darstellung von XML-Dateien im Quellcode, hierarchische Darstellung des Regelaufbaus etc., wurden bewusst weg gelassen.
Bis zum Projektende (Mai 2014) ist in diesem Client noch keine Funktionalität realisiert worden, die Benachrichtigungen so aufbereitet, dass eine Kontrolle durch den Benutzer möglich ist. Dieses wird in einem anderen Projekt im P23R-Umfeld, bei dem ein vollständiger Client implementiert werden soll, realisiert werden. Der im

[6] BUBE (Betriebliche Umweltdatenberichterstattung) ist ein Online-Werkzeug für die Berichterstattung gemäß 11. BImSchV, 13. BImSchV und PRTR-Gesetz.

Projekt entwickelte Client ist in der bestehenden Form ein nützliches Werkzeug, um Regeln testweise auszulösen.

E.4.2. Ergebnisse

Im Rahmen der Regelerstellung für weitere Berichtspflichten wurden eine Reihe von Modellen und Programmcodes erstellt, die unter freien Lizenzen allgemein verfügbar sind und von der Projektseite http://www.p23r4flex.de/ergebnisse herunter geladen werden können:

- Der Paketierer, der aus XML-Schemata, XSLT-Transformationen und anderen Rohdokumenten eine Regel erstellt und in eine Leitstelle hochlädt.
- Das Zwischenformat XUGFA kompakt
- ein Quellkonnektor, der aus einer eXist-Datenbank XUEmission auslesen und in XUGFA wandeln kann
- Erweiterungen der Musterimplementation, die verschiedene Abbrüche mit sinnvolleren Logmeldungen versehen sowie eine Art Ablaufplan der Regelauswertung angeben.
- Eine funktionale P23R-Regel im Format des neuen Paketierers, die die Berichterstattung nach 13. BImSchV ausgehend von XUEmission ermöglicht.
- Ein CommunicationConnector, der eine Benachrichtigung an BUBE Online hochladen kann
- Ein für die 13. BImSchV geeigneter P23R-Client mit rudimentärer Benutzerverwaltung und der Funktionalität, Regeln zu laden, aktualisieren und auszulösen.

E.5. Handbuch für die Regelerstellung im P23R

Aufbauend auf der Regelerstellung der beiden Berichtsprozesse wurde im Projekt ein Handbuch zur Regelerstellung entworfen. In einem internen Workshop mit der Firma ENDA und dem UBA wurde das Handbuch überarbeitet, Fehler eliminiert und in einer ersten Version finalisiert. Das in den nächsten Unterkapiteln eingefügte Handbuch stellt keine Endversion dar. In folgenden Projekten soll das Handbuch überarbeitet und den Veränderungen angepasst werden.

E.5.1. Einleitung

Dieser Leitfaden richtet sich an Regelersteller und Projektleiter, die P23RBenachrichtigungsregeln entwickeln oder entwickeln lassen wollen und einen praktischen Einblick benötigen. Er setzt voraus, dass der Leser mit den grundlegenden P23R-Konzepten ansatzweise vertraut ist und die P23R-Rahmenarchitektur sowie die Spezifikation der Technischen Benachrichtigungsregelsprache (T-BRS) gelesen oder zumindest einen kurzen Einblick genommen hat. Insbesondere enthält Abschnitt 2.1 der T-BRS einen vollständigen Überblick über den P23R.

 In diesem Einsteigerleitfaden wird eine Benachrichtigungsregel anhand eines kurzen Beispiels im P23R-Entwicklerportal erstellt und getestet. Dazu ist die Verwendung von Chrome zurzeit erforderlich. Die Unterstützung weiterer Browser ist geplant. Benachrichtigungsregeln bzw. -regelgruppen, die Pivot- Teildatenmodelle und Testsätze können auch mit allgemein verfügbaren, XML-unterstützenden Werkzeugen entwickelt werden. Die Vorgehensweise ist praktisch identisch. Allerdings empfiehlt es sich, zumindest die Vorlagen für Benachrichtigungsregelgruppen, Teildatenmodelle und Testgruppen sowie die Formular-Editoren des P23R-Entwicklerportals für die Manifest- Dateien zu nutzen.
Ziel des Leitfadens ist es, die Regelerstellung anhand eines kurzen fiktiven Praxisbeispiels zu erklären. Ein allgemeines Vorgehen wird nicht vorgestellt. Das Beispiel-Szenario wurde so gewählt, dass eine Berichtspflicht beschrieben wird, aus der eine kurze und übersichtliche Benachrichtigungsregel mit wichtigen Merkmalen von Benachrichtigungsregeln entwickelt werden kann. In diesem Beispiel soll ein Unternehmen im Rahmen eines Umzuges automatisch seine neue Adresse an die zuständige Stelle melden. Die fertige Benachrichtigungsregelgruppe kann hier (`https://entwickler.p23r.de/img/starter.zip`) heruntergeladen werden.
Bei der nachfolgenden Erläuterung des Beispiels werden der grundsätzliche Aufbau und die wichtigsten Bestandteile (Artefakte) einer Benachrichtigungsregel gezeigt. Die Entwicklung wird mithilfe des P23R-Entwicklerportals erläutert, welches in erster Linie zur Veröffentlichung und dem Testen von P23R-Benachrichtigungsregelgruppen, P23R-Pivot-Teildatenmodellen und P23R-Testsätzen dient. Außerdem kann es zur Erstellung und Bearbeitung von Artefakten, bspw. einer Benachrichtigungsregelgruppe, genutzt werden. Grundsätzlich können Regelersteller die Entwicklung der Artefakte auch mithilfe ihrer eigenen gewohnten XML-Werkzeuge durchführen, was hier aber nicht weiter betrachtet wird. Die Regelerstellung entspricht im Grunde genommen der Entwicklung von Software, d. h. es können typische Vorgehensmodelle wie das V-Modell XT angewendet werden. Dieser Leitfaden beschränkt sich auf die für eine P23R-Regelerstellung spezifischen Aspekte eines solchen Vorgehens. In diesem Leitfaden werden Verweise auf andere **Kapitel** fett und *Fachbegriffe* und *Objekte* kursiv dargestellt. Referenzen auf Elemente in Bildern können im Text mit *A#, B#* etc. erfolgen.
Im **Info**-Bereich des P23R-Entwicklerportals unter **Leitfäden** befindet sich eine **Checkliste**, die auflistet, welche Aktivitäten bei der Regelerstellung typischerweise durchgeführt werden müssen und welche Ergebnisse nach einer Aktivität vorliegen sollten. An dieser Checkliste orientiert sich das hier vorgestellte Beispiel. Neben dem Leitfaden befindet sich dort ebenso ein **Glossar**.

E.5.2. Anforderungsanalyse und Fachkonzept

Zunächst wird ein beispielhaftes Vorgehen zur Analyse einer Fachdomäne und der Gestaltung eines Fachkonzeptes vorgestellt. Die Meldung von Unternehmensumzügen dient dabei als fiktives Beispiel, um die Erhebung der Anforderungen darzustellen. Ebenfalls erfolgt an dieser Stelle die Vorstellung der Anforderungsanalyse und des Fachkonzeptes, um die sich anschließende Implementierung verständlich zu machen.

E.5.2.1. Grundlegende Anforderungsanalyse

Bevor eine Benachrichtigungsregelgruppe mit ihren Benachrichtigungsregeln erstellt werden kann, wird wie bei allen IT-Projekten ein Fachkonzept benötigt. Die Anforderungen der Benachrichtigungsregel entstammen der umzusetzenden Norm (Gesetz, Vorschrift etc.), die eine Melde- oder Berichtspflicht definiert. Daher soll in der für die Benachrichtigungsregelgruppe zu erstellenden Dokumentation im Abschnitt Anforderungen zuerst die Norm mit den für die Benachrichtigungsregelgruppe relevanten Textabschnitten zitiert werden, um die Ziele der zu erstellenden Benachrichtigungsregelgruppe zu dokumentieren.

```
§ 123 Unternehmensmeldepflichten bei Umzügen


... "Ein Unternehmen muss bei einem Umzug seine neue Adresse an die zuständige Stelle
innerhalb von 10 Werktagen elektronisch melden." ...


... "Die Eintragung der neuen Adresse wird durch die zuständige Stelle innerhalb von 10
Werktagen elektronisch bestätigt." ...
```

Eine Norm wird in der Regel durch Ausführungsvorschriften, Rundschreiben, verwandte Normen, Kommentierungen, Urteile etc. ergänzt. Die relevanten Textpassagen sollen analog in den Anforderungen dokumentiert werden.

```
Rundschreiben vom 01.04.2000


"Die alte und die neue Adresse, sowie die Bestätigung der Adressänderung sind im XML-Format
zu übermitteln. Der Kommunikationsweg wird durch die zuständige Stelle vorgegeben."
```

Weiterhin sollen Referenzen auf Normen, Benachrichtigungsregelgruppen, Projekte und Lösungen mit gleichen oder ähnlichen Aufgaben in den Anforderungen aufgeführt werden. Eine Analyse dieser Stoffsammlung soll bei Bedarf das Fachkonzept durch weitere fachliche und technische Anforderungen ergänzen. So kann wie im folgenden Beispiel dargestellt das Format der Adressen auf einen der verfügbaren Standards (hier: UN-CEFACT) festgelegt werden.

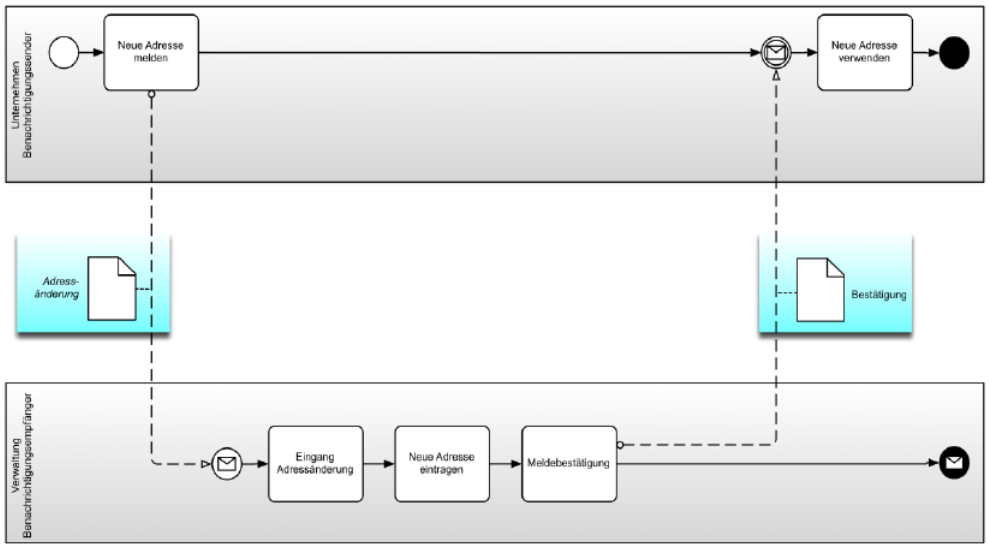
```
Anf-001 Format von Adressen
```

```
Das Format der zu übermittelnden Adressen SOLL sich an dem Format der UN-CEFACT Core-
Modelle orientieren.
```

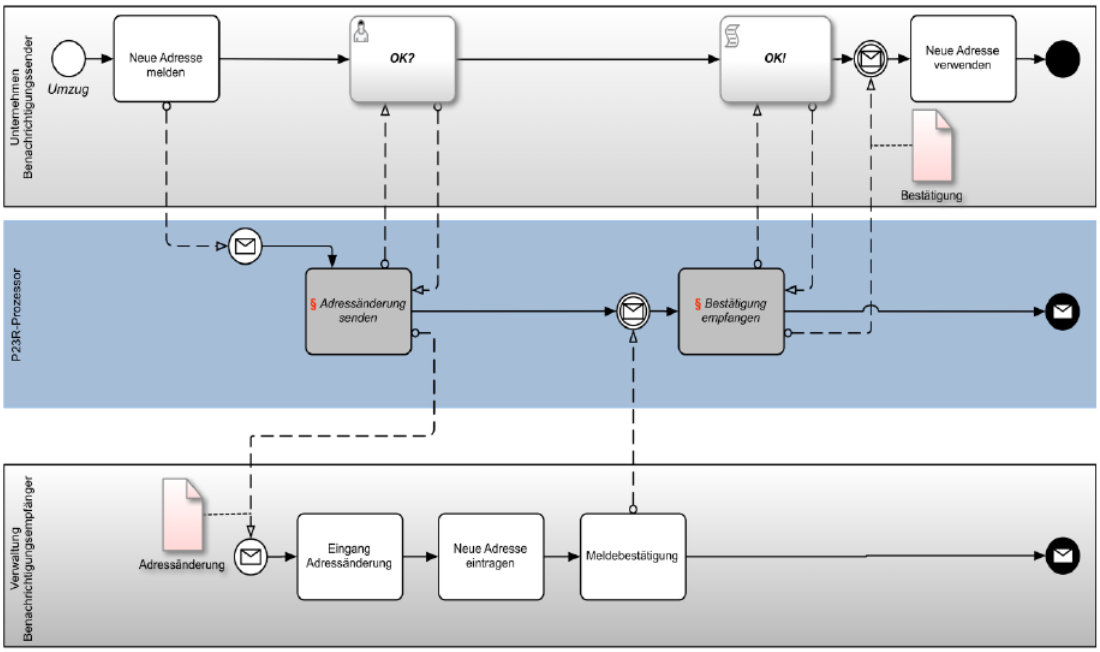
Für jede Anforderung muss ihre Verbindlichkeit (optional, empfohlen, verpflichtend) durch die Nutzung der Hilfsverben KANN, SOLL und MUSS sowie SOLL NICHT, DARF und DARF NICHT klar festgelegt werden. Außerdem darf eine Anforderung auch immer nur genau eine Anforderung beschreiben.
Die Benachrichtigungsregelgruppe ist typischerweise identisch mit der umzusetzenden Melde- und Berichtspflicht bzw. der Norm und wird in diesem Beispiel als *Unternehmensumzug* bezeichnet. Die fachliche Analyse soll als erstes Ergebnis die benötigten Benachrichtigungsregeln für die Benachrichtigungsregelgruppe liefern. Dazu müssen die Prozessketten zwischen den beteiligten Partnern - Unternehmen und Ver-

waltung - modelliert werden.

Für die übersichtliche Darstellung von Prozessketten soll bei P23R einheitlich das "Business Process Model and Notation" (BPMN) verwendet werden. Eine einheitliche Modellierung in BPMN hat zum Vorteil, dass Anforderungsanalysen konsistent und vergleichbar sind. Die Wahl des Modellierungswerkzeuges zur Erstellung von BPMN-Diagrammen ist dem Entwickler freigestellt. Die Erweiterung von Modellierungswerkzeugen mit frei verfügbaren Stencils zur Unterstützung von BPMN ist in der Regel möglich. Um die Prozesse detaillierter zu modellieren, können zusätzlich UML2 Modellierungsmethoden und –Werkzeuge genutzt werden.
Das nachfolgende vereinfachte Diagramm für die Unternehmensmeldepflichten bei Umzügen zeigt auf, dass zwischen dem Unternehmen und der Verwaltung einerseits eine Adressänderung und andererseits eine Bestätigung versendet werden. Diese beiden *Benachrichtigungen* zeigen an, dass zwei Benachrichtigungsregeln in der Benachrichtigungsregelgruppe benötigt werden. Der Detailierungsgrad eines Prozessketten-Diagramms soll möglichst knapp sein, da es vor allem auf den Austausch von Informationen zwischen den Beteiligten und weniger auf deren interne Tätigkeiten ankommt.



Für die P23R-spezifische Dokumentation einer Benachrichtigungsregelgruppe - hier *Unternehmensumzug* - wird anschließend das nachfolgende Diagramm eingefügt. Jede Benachrichtigung im Prozessketten-Diagramm wird durch den generischen P23R-Prozessbaustein ("*Maikäfer*", vgl. Abb. 3 der Rahmenarchitektur) für eine Benachrichtigungsregel ersetzt, d. h. im konkreten Fall durch *§Adressänderung senden* und *§Bestätigung empfangen*. Die Benachrichtigungsregeln werden in einer eigenen BPMN-Swimlane P23R-Prozessor gezeichnet, wobei dieser im hier dargestellten Beispiel zum Unternehmen gehört.



Nachdem geklärt ist, welche Benachrichtigungsregel für die Benachrichtigungsregelgruppe benötigt werden, muss ermittelt werden, welche Datenmodelle gebraucht werden. Jede Benachrichtigungsregel hat eine oder mehrere Datenquellen, aus der eine Benachrichtigungsregel Daten abfragen darf, die zur Generierung der zu versendenden Benachrichtigung benötigt werden. Typischerweise gibt es interne Datenquellen, wie die fachspezifischen IT-Systeme, mit denen das Unternehmen arbeitet, die primär als Datenquellen für den P23R-Prozessor dienen. Daneben kann es aber auch externe Datenquellen geben, beispielsweise im Internet, auf die der P23R-Prozessor zugreifen darf. Der P23R-Prozessor greift auf die Datenquellen immer über einen Quelldatenkonnektor zu, der Teil der P23R-Infrastruktur im Unternehmen ist. Darüber hinaus gibt es öffentlich zugängliche Daten, die über öffentliche Quelldatenkonnektoren, wie beispielsweise der bei der Öffentlichen P23R-Leitstelle, über den die Kommunikationsadressen und -parameter der Kommunikationsendpunkte zum Übertragen der fertigen Benachrichtigung abgefragt werden. Um die Benachrichtigungsregeln realisieren zu können, müssen die benötigten Daten der Datenquellen analysiert werden:

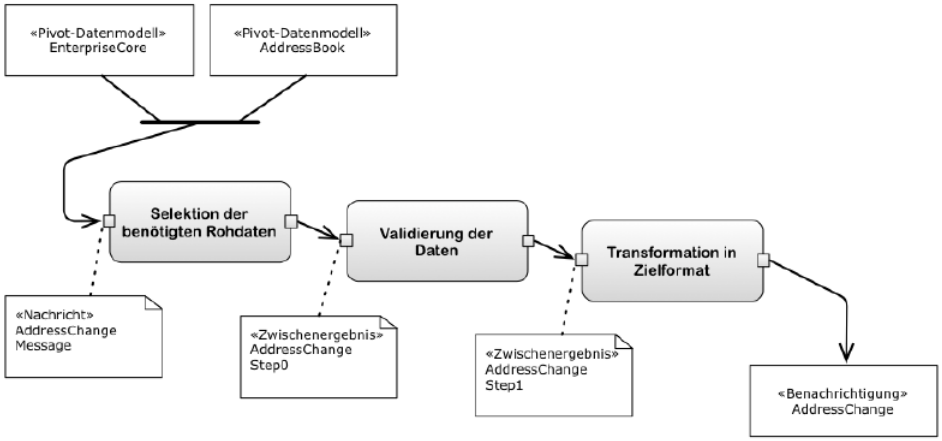
- Über welchen Quelldatenkonnektor und mit welchem Pivot-Datenmodell sind die Quelldaten (Rohdaten) verfügbar?
- Wie sieht das Datenformat (Datenmodell) der zu versendenden Benachrichtigungen aus?

Nachdem die benötigten Datenmodelle ermittelt wurden, sollte abschließend überlegt werden, welche Aufgaben die einzelnen Benachrichtigungsregeln umsetzen sollen, beispielsweise:

- syntaktische Überprüfung der Quelldaten
- semantische Überprüfung der Daten, in dem Beispiel etwa, ob eine Adresse überhaupt existiert

- Berechnung der zusätzlichen Werte durch Ableitung, bspw. die Anzahl der Mitarbeiter eines Unternehmens
- Transformation in das Zielformat
- etc.

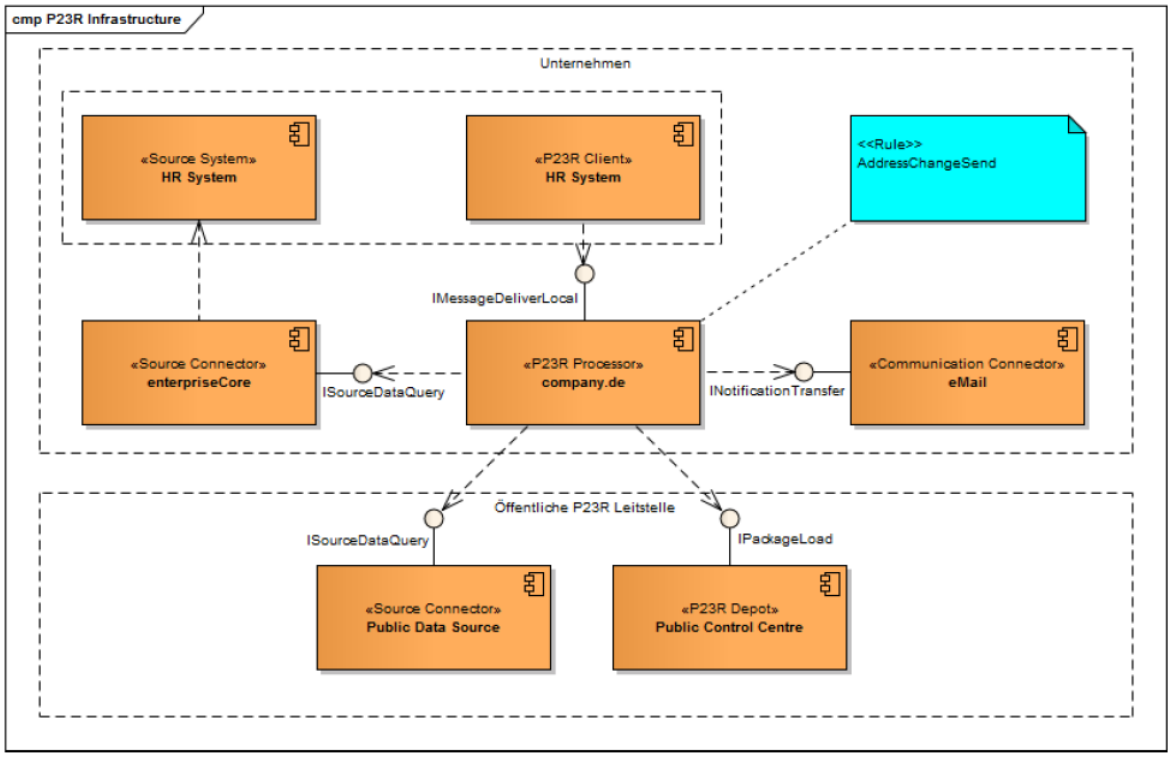
Aus dem Umfang an Aufgaben ergeben sich die konkreten Schritte, die in der Benachrichtigungsregel implementiert werden sollen.



Bei Bedarf sollen die fachlichen und technischen Anforderungen an die jeweiligen Benachrichtigungsregeln ergänzt und dokumentiert werden.

E.5.2.2. Systemabgrenzung

Bevor eine Benachrichtigungsregel implementiert wird, sollte die zur Umsetzung benötigte P23R-Infrastruktur aus Sicht des Unternehmens als UML-Komponentendiagramm dokumentiert werden. Nachfolgend ist ein Beispiel für die Benachrichtigungsregel *§Adressänderung* senden (AddressChangeSend) dargestellt.



Den Kern der lokalen P23R-Infrastruktur (Unternehmen) bildet immer der lokale P23R-Prozessor. Um auf die für die Benachrichtigungsregel benötigten Daten zugreifen zu können, müssen die benötigten Quelldatenkonnektoren (SourceConncetor) identifiziert werden. In diesem Beispiel fragt der P23R-Prozessor die alte und neue Adresse des Unternehmens vom internen HR-System ab, wobei es das Pivot-Teildatenmodell enterpriseCore nutzt. Mit den abgefragten Daten wird die Transformation entsprechend der Benachrichtigungsregel durchgeführt und die Benachrichtigung generiert. Dabei benutzt die Benachrichtigungsregel Daten der Öffentlichen P23R-Leitstelle mittels des Öffentlichen Quelldatenkonnektors, in diesem Beispiel die Adressen aus dem Pivot-Teildatenmodell AddressBook.
Die fertigen Benachrichtigungen werden mithilfe eines Kommunikationskonnektors (CommunicationConnector) übertragen. In diesem Beispiel erfolgt die Übertragung per E-Mail durch den Kommunikationskonnektor eMail.
Die Auslösung der Generierung einer Benachrichtigung wird mittels eines P23R-Client durchgeführt. In diesem Beispiel wird davon ausgegangen, dass die Oberfläche für das Auslösen und die Freigabe direkt in ein HR-System integriert sind.
Falls weitere Anforderungen für die Implementierung und Integration der Quelldaten- und Kommunikationskonnektoren feststehen, sind diese zu ergänzen. Bspw. können für die hier entwickelte Benachrichtigungsregel die folgenden Anforderungen ergänzt werden:

```
Anf-002 Übertragung der Benachrichtigungen
Das Senden der Adressänderung und der Empfang der Bestätigung MUSS per E-Mail mit dem
Protokoll _SMTPS_ erfolgen.



Anf-003 Sichere Übertragung der Benachrichtigungen


Das Senden der Adressänderung und der Empfang der Bestätigung MUSS vertraulich durch
Verschlüsselung des Inhalts der E-Mail erfolgen.
```

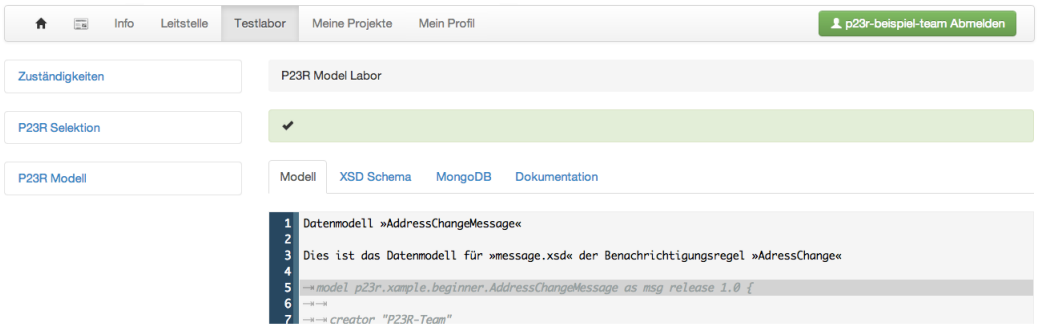
E.5.3. Modellierung der Datenmodelle

Bevor mit der eigentlichen Implementierung einer Benachrichtigungsregel gestartet werden kann, müssen die ermittelten Datenformate genau spezifiziert werden:

- das Datenformat für die zu erzeugende und zu versendende Benachrichtigung,
- die Pivot-Teildatenmodelle, d. h. die Datenformate der Rohdaten, sowie
- das Datenformat der eingehenden Nachricht, die die Generierung der Benachrichtigung auslösen und starten soll.

In einer Benachrichtigungsregel sind alle Datenformate als XSD-Schemata zu spezifizieren. Solch ein XSD-Schemata kann beispielsweise mit grafischen Entwicklungswerkzeugen oder einem Texteditor erzeugt werden.

In diesem Beispiel werden die Möglichkeiten des P23R-Entwicklerportals genutzt, die XSD-Schema in P23R Model, einer domänenspezifischen Sprache, zu formulieren. Durch übersichtlichere und verständlichere Beschreibungen können so Datenmodelle beschrieben werden, die auch für eine Zielgruppe ohne technischen Hintergrund verständlich sind. Im P23R-Entwicklerportal werden*.model-Dateien automatisch in XSD-Schema-Dateien übersetzt. Außerdem können sie im Testlabor des P23R-Entwicklerportals ausprobiert werden. P23R Model wird im literate-programming verfasst, daher wird der Code immer mit einem Tab eingerückt. Zeilen ohne führendes Tab beinhalten Kommentare. Die resultierenden XSD-Schemata sind in der Archiv-Datei für die Benachrichtigungsregelgruppe enthalten.



E.5.3.1. Benachrichtigungsformat

Das Wichtigste vor der Implementierung einer Benachrichtigungsregel ist, zu klären, welche Daten in der Benachrichtigung enthalten sein müssen bzw. zur Erstellung benötigt werden. Da es für dieses Beispiel kein existierendes Datenformat gibt, beispielsweise ein XÖV-Standard oder eine fachgebietspezifische Vorgabe, wie Bausteine aus XUBetrieb im Bereich Umwelt, wird für dieses Beispiel ein eigenes Datenmodell in P23R Model erstellt. Dieses enthält für dieses Beispiel den eindeutigen Namen des Unternehmens, die aktuelle und die neue Adresse.

```
Dies ist ein P23R Model. Der Quellcode des Models muss mit einem Tab eingerückt sein.


    model P23R.ruleGroups.Starter.AddressChange.Notification as ntf release 1.0 {


        creator 'P23R-Team'

        organisation 'P23R-Team'

        base http://leitstelle.p23r.de/NS/

        depends on p23r release 1.0


        title 'Format für die Benachrichtigung einer Adressänderung'


        tag 'example'

        tag 'notification'

        tag 'P23R-Regeln für Einsteiger'


        description 'Dieses Model beschreibt die zu versendenen Daten bei der
Adressänderung eines Unternehmens entsprechend §123 Unternehmensmeldepflichten bei
Umzügen.'
```

```
        namespace cc=http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/CoreComponents1-0
file "rule:///enterpriseMove/CoreComponents.xsd"


        entity Notification {
            companyName : String
            currentAddress : cc:Address
            newAddress : cc:Address
        }
    }
```

Entsprechend den Anforderungen werden für die Modellierung der Adresse die Bausteine der UNCEFACT Core Components (`http://www.unece.org/cefact/codesfortrade/uncc1/ccl_index.html`) verwendet. Diese seien nachfolgend beispielhaft spezifiziert.

```
Dies ist ein P23R Model. Der Quellcode des Models muss mit einem Tab eingerückt sein.


    model P23R.ruleGroups.Starter.CoreComponents as cc release 1.0 {


        creator 'P23R-Team'
        organisation 'P23R-Team'
        base http://leitstelle.p23r.de/NS/
        depends on p23r release 1.0


        title 'Formate fÃ¼r die UN/CEFACT Core Components'


        tag 'example'
        tag 'core components'
        tag 'P23R-Regeln für Einsteiger'


        description 'Das Model beschreibt die benötigten Datenformate (bcc:address) der
UN/CEFACT Core Components.'


        entity Address {
            identification: Identifier optional
            streetName: String
```

```
            buildingNumber: String
            cityName: String
            postcode: Identifier
            country: Identifier optional
        }

        entity Identifier: string
    }
```

E.5.3.2. Pivot-Teildatenmodelle

Nachdem festgelegt ist, welche Daten benötigt werden und im Rahmen der Systemabgrenzung geklärt wurde, in welchem Quellsystem die Daten verwaltet werden und welche Quelldatenkonnektoren genutzt werden, müssen die zugehörigen Pivot-Teildatenmodelle der Quelldatenkonnektoren die notwendigen Rohdaten liefern. Falls in einer Domäne bereits Regeln und Pivot-Teildatenmodelle erstellt wurden, sollen die bestehenden Pivot-Teildatenmodelle wiederverwendet und bei Bedarf ergänzt werden.
Im Folgenden wird ein eigenes Pivot-Teildatenmodell beispielhaft definiert.
Jedes Unternehmen hat eine Adresse für ihren Hauptsitz. Diese kann sich im Laufe der Zeit beispielsweise durch Umzüge ändern. Das Pivot-Teildatenmodell mit den grundlegenden Informationen über ein Unternehmen muss daher die Adressen von Standorten historisiert bereitstellen.
Das nachfolgende Modell umfasst nur die im Beispiel benötigten Daten.
Während das Zielformat hauptsächlich durch den Empfänger und dessen Anforderungen festgelegt wird, richtet sich das Pivot-Teildatenmodell sowohl nach dem Bedarf der Regelersteller als auch den Anforderungen und Gegebenheiten der Hersteller der IT-Fachsysteme. Dementsprechend können die Datenmodelle voneinander abweichen, in diesem Beispiel die Repräsentation von Adressen.

```
Dies ist ein P23R Model. Der Quellcode des Models muss mit einem Tab eingerückt sein.

    model P23R.ruleGroups.Starter.enterpriseCore as ec release 1.0 {

        creator "P23R-Team"
        organisation "P23R-Team"
        base http://leitstelle.p23r.de/NS/
        depends on p23r release 1.0

        title "Format für das Pivot-Teildatenmodell enterpriseCore"
```

```
        tag "example data model"
        tag "beginner"
        description "Dieses Datenmodell hält die aktuellen und vorherigen Adressen eines
Unternehmens bereit."

        entity EnterpriseCore {
            profile: Profile
        }

        entity Profile {
            company: Company
            headQuarter: Location many
        }

        entity Company {
            name: string
        }

        entity Location {
            address: Address
            validFrom: date
            lastModified: date
        }

        entity Address {
            houseNumber: string
            street: string
            city: string
            zipCode: string
            country: string
        }
    }
```

Um Adressen auf ihre Gültigkeit zu überprüfen, kann eine Adresse beim Öffentlichen Quelldatenkonnektor abgefragt werden. Wenn die Adresse dort bekannt ist, wäre die Adresse gültig. Das nachfolgende Pivot-Teildatenmodell beschreibt das notwendige Schema für diesen Zweck.

```
Dies ist ein P23R Model. Der Quellcode des Models muss mit einem Tab eingerückt sein.

    model P23R.cc.addressBook as ab release 1.0 {

        creator 'P23R-Team'
        organisation 'P23R-Team'
        base http://leitstelle.p23r.de/NS/
        depends on p23r release 1.0

        title 'Format für das Pivot-Teildatenmodell addressBook'

        tag 'example'
        tag 'address book'
        tag 'P23R-Regeln für Einsteiger'

        description 'Dieses Datenmodell beschreibt das Format des Adressbuchs mit allen
gültigen, postalischen Adressen, die der Öffentliche Quelldatenkonnektor der P23R-
Leitstelle kennt.'

        namespace cc=http://leitstelle.p23r.de/NS/ruleGroups/Starter/CoreComponents file
"model:///addressBook/CoreComponents.xsd"

        entity AddressBook {
            addresses: cc:Address many optional
        }
    }
```

E.5.3.3. Nachrichtenformat

Wenn eine Benachrichtigungsregel zur Generierung der § Adressänderung ausgelöst werden soll, muss ein XML-Dokument mit dem XSD-Schema der Message.xsd und deren Namespace vom P23R-Prozessor empfangen werden.
Nachfolgend ist das konkrete XSD-Schema Message.xsd für die eingehende Nachricht aufgeführt. Diese muss weitestgehend der Vorgabe für Message.xsd entsprechen. Da keine benachrichtgungsregelspezifischen Erweiterungen benötigt werden, wurde nur der Namespace des XSD-Schemas für die Benachrichtigungsregel und der Standardwert für subject direkt im XSD-Schema angepasst.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Message.xsd -->

<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Message1-0"

targetNamespace="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Message
1-0"
  elementFormDefault="qualified"
  version="1.0">
  <xsd:complexType name="P23RMessageType">
    <xsd:sequence>
      <xsd:element name="messageId" type="xsd:string"/>
      <xsd:element name="subject" type="xsd:string" default="«rule name»" minOccurs="0"/>
      <xsd:element name="transactionId" type="xsd:string" minOccurs="0"/>
      <xsd:element name="applyAt" type="xsd:date" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
  </xsd:complexType>
  <xsd:element name="message">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="P23RMessageType">
          <xsd:sequence>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

E.5.4. Anlegen von Projekten im P23R-Entwicklerportal

Nachdem die konzeptionellen Vorarbeiten geleistet wurden, kann die eigentliche Implementierung im P23R-Entwicklerportal beginnen. Für die Benutzung des Portals

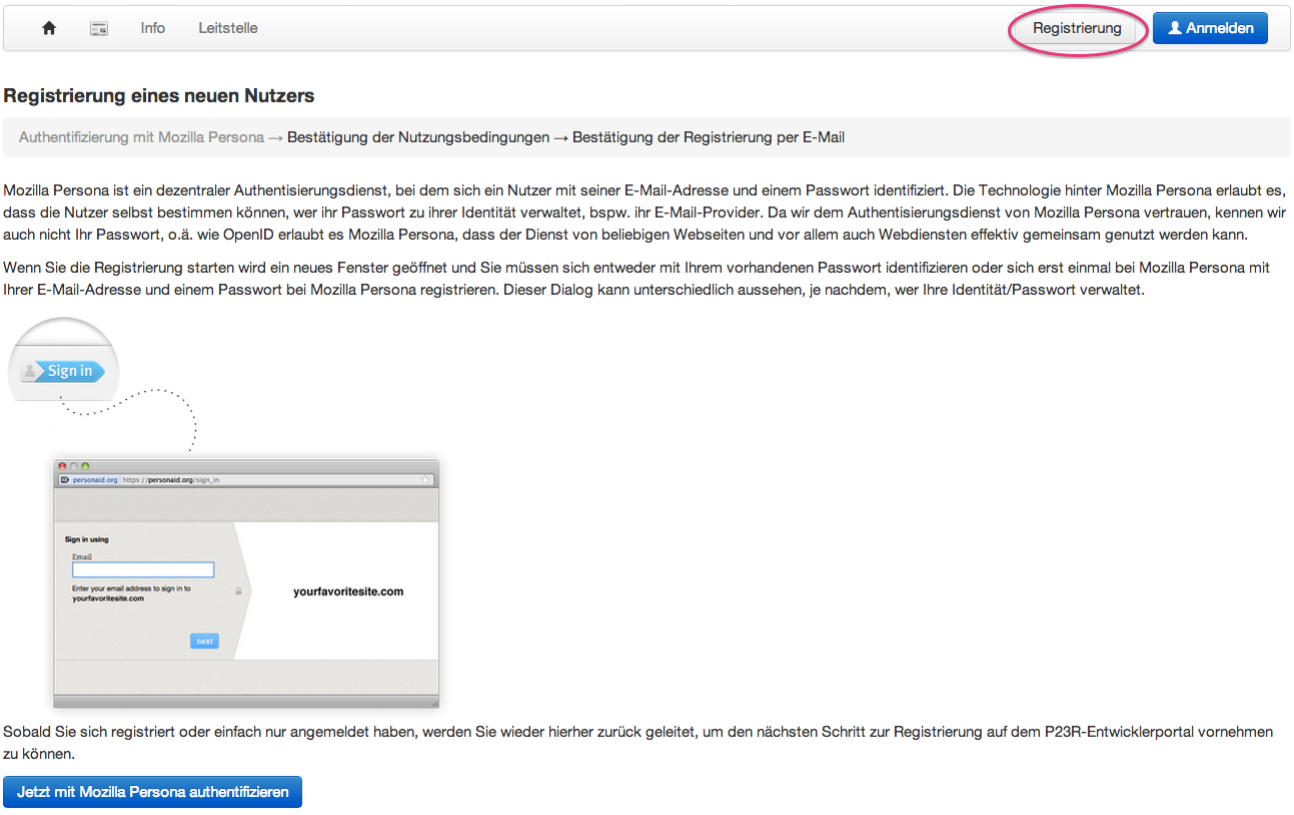
wird die Verwendung eines aktuellen Chrome-Browsers empfohlen, da andere Browser noch nicht umfassend getestet wurden und erforderliche Anpassungen erfolgt sind. Das P23R-Entwicklerportal dient den Regelerstellern und Datenmodellierern vorrangig zum:

- Hochladen von Benachrichtigungsregelgruppen, Pivot-Teildatenmodellen, Testsätzen und Zuständigkeiten in das P23R-Entwicklerportal,
- Veröffentlichen von Benachrichtigungsregelgruppen, Pivot-Teildatenmodellen und Testsätzen in Projekt-Leitstellen,
- Überprüfen und Entwickeln von Benachrichtigungsregelgruppen, Pivot-Teildatenmodellen, Testsätzen und Zuständigkeiten,
- Veröffentlichen von Benachrichtigungsregelgruppen, Pivot-Teildatenmodellen, Testsätzen und Zuständigkeiten in der Öffentlichen Leitstelle
- Testen von Benachrichtigungsregelgruppen

Dazu müssen Benachrichtigungsregelgruppen, Pivot-Teildatenmodelle, Testsätze und Zuständigkeiten und die Projekt-Leitstelle in Form von einzelnen Projekten angelegt werden. Die folgenden Schritte sind für die Regelerstellung mit dem P23R-Entwicklerportal notwendig.

E.5.4.1. Registrierung

Damit ein Regelersteller, Datenmodellierer etc. Projekte anlegen und bearbeiten kann, muss er sich einmalig im P23R-Entwicklerportal registrieren.



Für die Nutzung des P23R-Entwicklerportals ist eine Kennung bei dem Authentifierungsdienst Mozilla Persona (`https://login.persona.org/`) notwendig. Falls noch kein Nutzer bei Mozilla Persona (`https://login.persona.org/`) vorhanden ist, muss sich dort registriert werden, da dieser unabhängige Dienst zur Autorisierung im P23R-Entwicklerportal genutzt wird. Ansonsten wird ein neuer Nutzer schrittweise durch die Registrierung am P23R-Entwicklerportal geführt und gleich angemeldet.

E.5.4.2. Einrichtung von Projekten

Bevor ein Projekt angelegt werden kann, wird ein GIT-Repository (`http://git-scm.com/`) benötigt, in dem alle Dateien des Projektes gespeichert und zwischen den Regelerstellern, Datenmodellierern etc. und dem P23R-Entwicklerportal in beiden Richtungen ausgetauscht werden können. Das GIT-Repository ist nicht Teil des P23R-Entwicklerportals, sondern muss durch die Entwickler selbst bereitgestellt werden.

Das bekannteste GIT-Repository ist beispielsweise GITHub, bei dem die Arbeitsweise von GIT-Repositories gut veranschaulicht ist. Bei GITHub können OpenSource-Projekte kostenlose und damit öffentlich sichtbare GIT-Repositories erhalten. Darüber hinaus können bei GITHub und vergleichbaren Diensten GIT-Repositories gemietet werden. Die GITHub zugrundeliegende, web-basierte Software GITLab ist OpenSource und sollte auf einem eigenen Server zur Verwaltung von GIT-Repositories installiert werden, wenn sie in eigenen Projekten mit größeren Teams häufiger benötigt werden. Eine weitere Möglichkeit einen zentralen GIT-Server aufzusetzen ist gitolite. Dieser Server für Unix-Systeme erlaubt insbesondere die feine Einstellung von Zugriffsrechten.
Wenn beispielsweise ein handelsüblicher virtueller LINUX-Server gemietet wurde, auf dem über SSH-Nutzer zugegriffen werden kann, sollte für das GIT-Repository ein eigener Nutzer eingerichtet werden, der auf einen eingeschränkten Bereich des Dateisystems Zugriff hat. Mit den folgenden Befehlen wird ein GIT-Repository auf dem Server eingerichtet:

```
cd «Pfad zum GIT auf dem Server»

mkdir «Name des GITs».git

cd «Name des GITs».git

git init --bare
```

Weiterhin muss der vom P23R-Entwicklerportal benötigte Branch p23rDP im GIT-Repository bekanntgemacht werden. Dabei muss er auf einem lokalen Rechner erzeugt und im GIT-Repository - auch Bare-Repository genannt - auf dem Server gespeichert werden.
(A#) Dazu wird zuerst lokal ein GIT-Clone eingerichtet.
(B#) Um den Branch anlegen zu können, muss ein beliebiger Inhalt erzeugt werden.
(C#) Nun muss die Änderung mit einer entsprechenden Info über den Zweck der Änderung commited werden.

(D#) Abschließend kann der Branch lokal definiert, im GIT-Repository gespeichert und damit endgültig für alle bereitgestellt werden.

```
# A#

mkdir -p «Name des GITs»

git init

cd «Name des GITs»

git remote add origin «Server + Pfad zum GIT auf dem Server mit dem zentralen GIT-
Repository»/«Name des GITs».git


# B#

touch README.md


# C#

git add . README.md

git commit -a -m "initial commit"


# D#

git branch p23rDP

git checkout p23rDP
git push origin p23rDP
```

Dies muss für jedes Projekt einzeln durchgeführt werden.
Die Platzhalter «...» sind durch die entsprechenden Pfadnamen/Dateinamen zu ersetzen. Für dieses Beispiel würde man folgende GITs benötigen:

- starter-enterpriseCore (Pivot-Teildatenmodell)
- starter-addressBook (Pivot-Teildatenmodell)
- starter-enterpriseMove (Benachrichtigungsregelgruppe)
- starter-testset (Testsatz)
- starter-depot (Projektleitstelle)

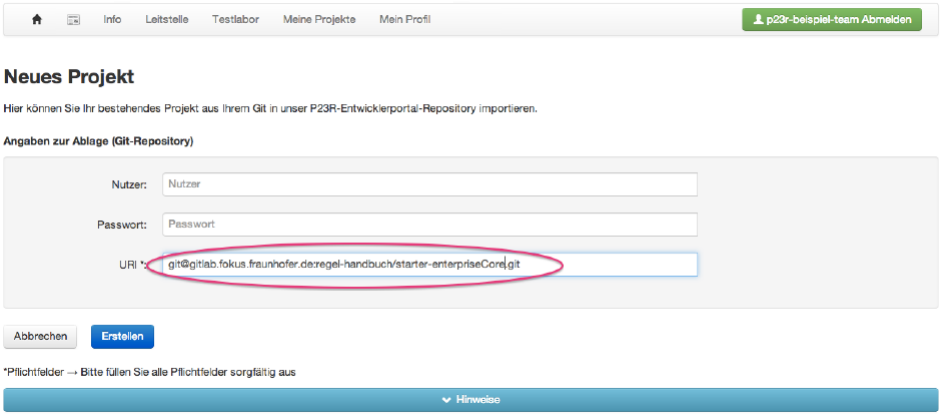
Darüber hinaus muss einmalig die Identität des P23R-Entwicklerportals dem SSH-Zugang für die lokalen GIT-Repositories bekanntgemacht werden. Dazu ist einmalig folgender Befehl auf dem Server notwendig:

```
cat <<EOF >> ~/.ssh/known_hosts

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC2NttcFE6r7P+WyU3Nwc26az0s0zMTWZLeXMNdE/rt9EQweqV/zbIpQiE1R2d
APYMHTqXNMzm93VmV/R7INz84Pp2tl3SMabgUcEkWnxKz4UT9V+w0Ed/0gOqqIWHruHBXaYvJRXBcL7h3W+y7Biq588
eZOmJX7b6HthEdKGPIcxBm0YuRq0MuuW+hVdz56HXriEUCa7FggRY9D/zgucmSsAp9rp7xsaYtrjE5kHSjW5/l5T2qe
0QmFnt+PVZ/arbEI5eGKW7Jy29it6wcr2JbDbtnJvQNXMurPePtTftGH7qMRzgFz28kT6PgIe6RWwDU1ugdb3JNDziV
nuJl25xt webmaster@p23r.de

EOF
```

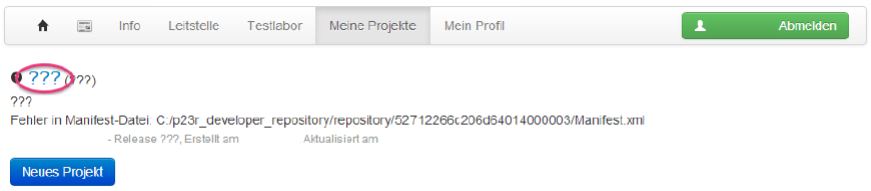
Weitergehende Informationen zum Einrichten des GIT-Repository befinden sich im FAQ-Bereich (`https://entwickler.p23r.de/info/`) des P23R-Entwicklerportals unterEntwicklerportal.
Wenn man das GIT-Repository erfolgreich eingerichtet hat, können neue Projekte im P23R-Entwicklerportal eingerichtet werden. Ein angemeldeter Benutzer im P23R-Entwicklerportal kann in die Ansicht Meine Projekte wechseln, um neue Projekte anzulegen. Ein Assistent führt durch die Erstellung des Projektes.
Es können leere oder bestehende GIT-Repositories importiert werden.
Dabei ist zu beachten, dass ein Branch p23rDP in beiden Fällen existieren muss.

E.5.5. Erstellung der Pivot-Teildatenmodelle

Wenn ein neues Projekt erzeugt wird, muss das entsprechende GIT-Repository angegeben werden, in dem die Daten abgelegt und ausgetauscht werden sollen. Bei URI ist der eigene Server/Pfad für das GIT-Repository einzutragen.
Nach dem Erzeugen des Projektes sind die Daten im Branch p23rDP sowohl im P23R-Entwicklerportal verfügbar als auch für die Entwickler extern zugänglich. Weitere Branches können im GIT-Repository existieren, werden aber vom P23R-Entwicklerportal nicht benutzt oder verändert, d. h. stehen den Entwicklern für ihre Arbeit exklusiv zur Verfügung und können weitere (geschützte) Daten enthalten.

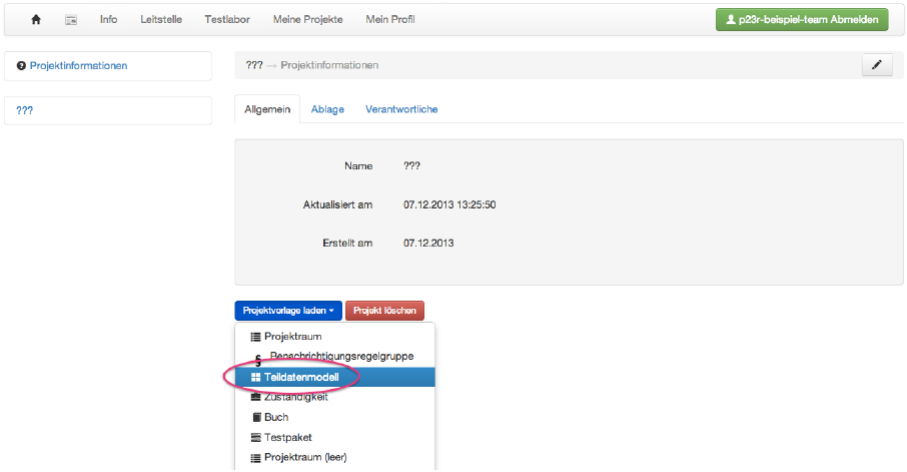


Das neue Projekt erscheint nun als leeres Projekt in der Liste der eigenen Projekte, wenn das GIT-Repository bisher noch leer war, d. h. keine Datei Manifest.xml enthält.



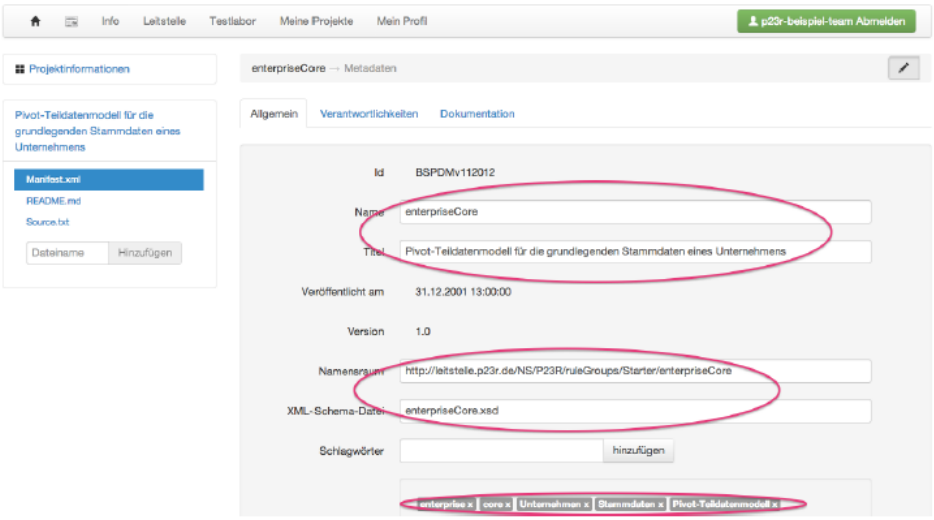
E.5.5.1. Projektvorlage

Nachfolgend wird das Projekt geöffnet und mittels einer Projektvorlage gefüllt, die alle Dateien enthält, die für einen Projekttyp üblicherweise benötigt werden. In diesem Beispiel wird mit der Projektvorlage Teildatenmodell gestartet.



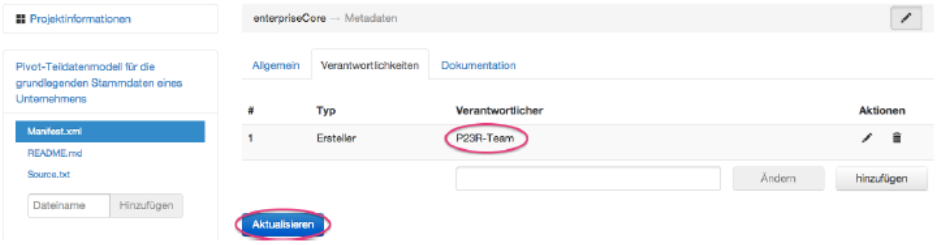
E.5.5.2. Anpassung des Manifestes

Die Vorlage für das Pivot-Teildatenmodell wird nun schrittweise angepasst. Zuerst wird links in den Navigation der Projektinhalte "Neues Modell" ausgewählt und das Manifest im Web-Formular angepasst.
Im Abschnitt Allgemein wird der gewünschte Name und Titel eingegeben.
Außerdem wir der Namespace übernommen und der Dateiname angegeben, unter dem das XSD-Schema im Pivot-Teildatenmodell abgelegt wird, d. h. typischerweise «Name des Pivot-Teildatenmodells».xsd.
Zuletzt wird das Pivot-Teildatenmodell mittels mehrerer Schlagwörter klassifiziert.



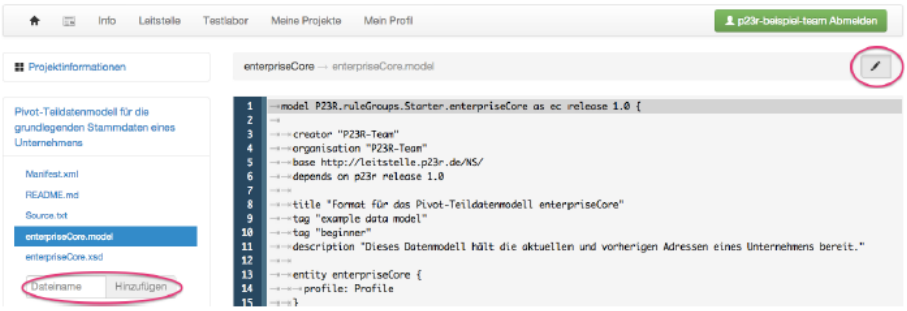
Anschließend wird der Abschnitt Verantwortlichkeiten bearbeitet, bei dem man den Namen des verantwortlichen Erstellers angeben muss. Der Abschnitt Dokumentation muss nicht geändert werden. Zuletzt werden die Änderungen mit "Aktualisieren"

gespeichert.

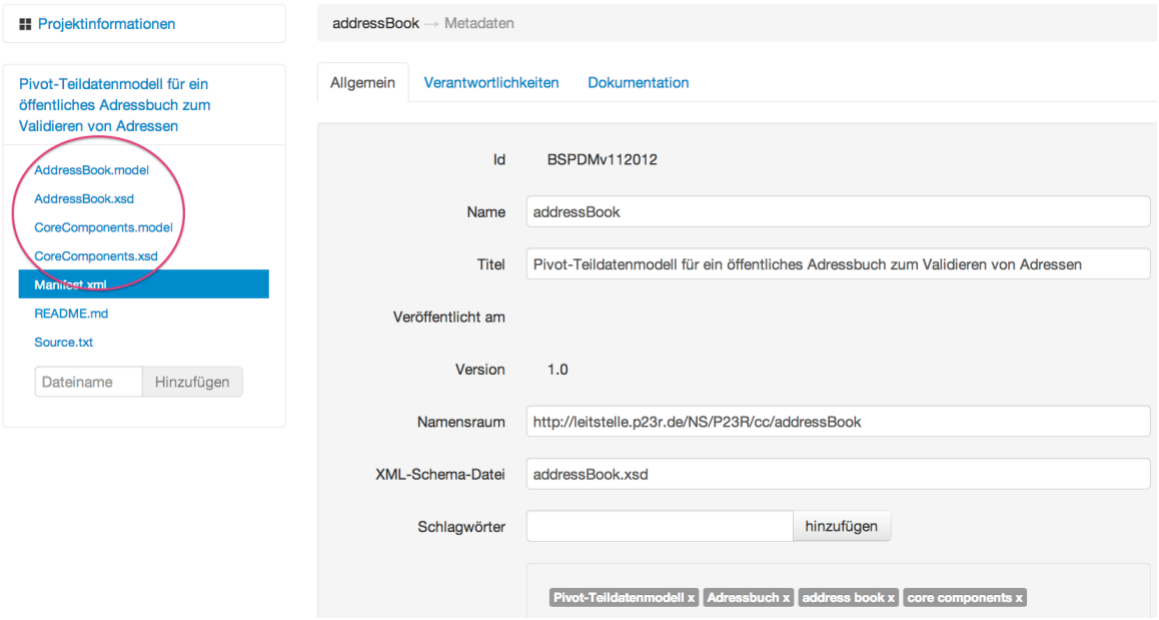


E.5.5.3. Einpflegen der Schemata

Zunächst wird eine neue Datei enterpriseCore.model hinzugefügt.
In die leere Datei wird der oben beschriebene Inhalt für enterpriseCore als P23R Model eingefügt. Dabei ist zu beachten, dass die Datei sich im Editor-Modus befindet (Stift-Symbol muss eingedrückt sein).
Abschließend wird die Datei mit "Aktualisieren" gespeichert. Mit dem Abspeichern der Datei wird diese auch automatisch in eine XSD-Schema-Datei enterpriseCore.xsd übersetzt.



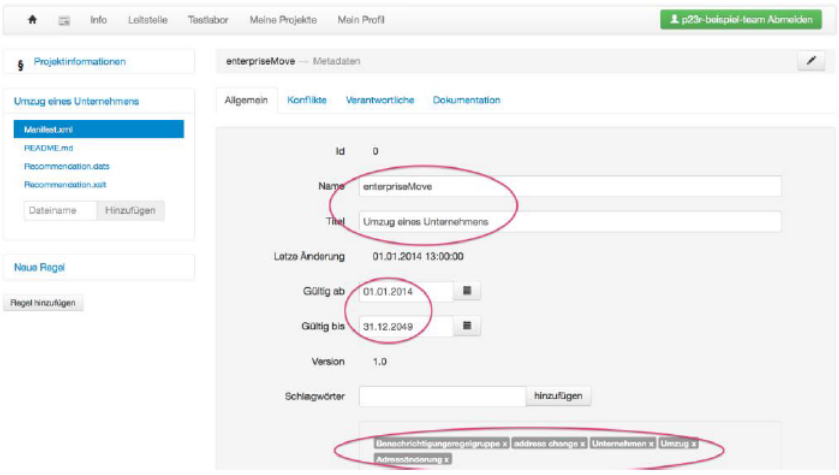
Um die Daten im P23R-Entwicklerportal oder im GIT-Repository zu aktualisieren, müssen diese am Ende explizit unterProjektinformationen synchronisiert werden.
Analog zum Pivot-Teildatenmodell enterpriseCore muss auch das Pivot-Teildatenmodell addressBook im P23R-Entwicklerportal eingepflegt werden. Dazu sind die obigen Schritte zum Anlegen von Projekten und eines Pivot-Teildatenmodells entsprechend zu wiederholen.
Da das Pivot-Teildatenmodell addressBook nicht nur das XSD-Schema AddressBook.xsd, sondern auch CoreComponents.xsd benötigt, muss zusätzlich auch das P23R Model CoreComponents.model erzeugt und übernommen werden.



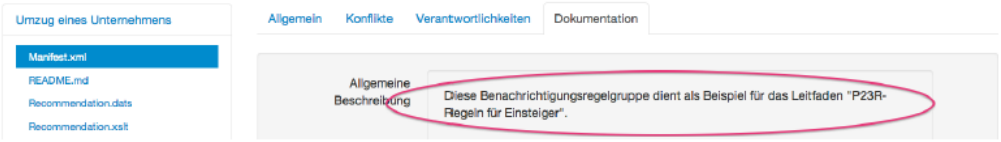
Nach Definition des Pivot-Teildatenmodells kann anschließend die eigentliche Benachrichtigungsregel realisiert werden.
Parallel können die Entwickler mit der Realisierung der Quelldatenkonnektoren zum Abruf der Unternehmensdaten anhand des Pivot-Teildatenmodells beginnen.

E.5.6. Erstellung der Benachrichtigungsregelgruppe

Zur Erstellung der Benachrichtigungsregelgruppe enterpriseMove für § 123 Unternehmensmeldepflichten bei Umzügen muss ein Projekt mit der Projektvorlage Benachrichtigungsregelgruppe erzeugt und das Manifest im Abschnitt "Allgemein" entsprechend ausgefüllt werden. Insbesondere muss auf eine eindeutige ID geachtet werden.
Der Abschnitt "Konflikte" muss nicht weiter geändert werden, da keine konfligierenden Benachrichtigungsregelgruppen bekannt sind.
Im Abschnitt "Verantwortlichkeiten" ist ein Ersteller einzutragen.



Im Abschnitt "Dokumentation" ist der Zweck der Benachrichtigungsregelgruppe kurz zu erläutern.



Die optionalen Skripte zur Konfiguration von persistenten Standardwerten Configuration.dats und Configuration.xslt werden für dieses Beispiel nicht benötigt und können gelöscht werden.
Die verpflichtenden Skripte Recommendation.dats und Recommendation.xslt zur Empfehlung einer Benachrichtigungsregelgruppe dürfen nicht gelöscht werden. Das Skript Recommendation.dats erzeugt gemäß der P23R-Spezifikation eine XML-Datei mit statischem Inhalt, der grundsätzlich empfiehlt, die Benachrichtigungsregelgruppe zur Aktivierung dem Verwalter der Benachrichtigungsregeln anzuzeigen. In dem Beispielszenario ist keine Anpassung der Recommendation-Skripte notwendig.

```
recommendationResult {

    @namespace { "http://leitstelle.p23r.de/NS/p23r/nrl/RecommendationResult1-0" }

    @recommended { true }

}
```

Da keine weiteren Berechnungen und Transformationen zur Generierung einer Empfehlung notwendig sind, kopiert Recommendation.xsltmit dem folgenden XSLT-Code das Ergebnis der Recommendation.dats:

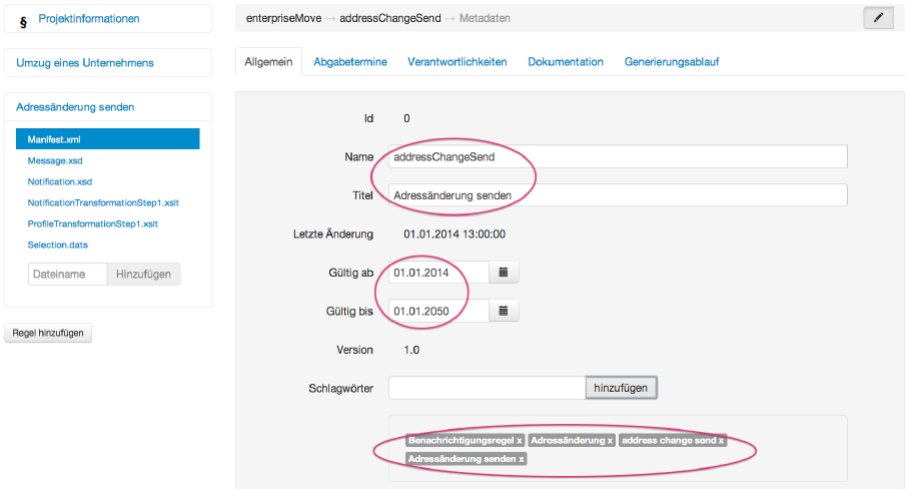
```
<?xml version="1.0" encoding="UTF-8"?>


<!-- Recommendation.xslt -->


<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>


  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>


</xsl:stylesheet>
```

Damit wurde die Benachrichtigungsregelgruppe erstellt, welche zwischendurch unter "Projektinformationen" synchronisiert werden sollte, bevor mit der Erstellung der Benachrichtigungsregel fortgefahren wird.

E.5.7. Erstellung der Benachrichtigungsregel

E.5.7.1. Manifest der Benachrichtigungsregel

Mit der Erstellung einer neuen Benachrichtigungsregelgruppe wird gleichzeitig eine Benachrichtigungsregel Neue Regel angelegt.
Die optionalen Skripte dieser Benachrichtigungsregel, werden für dieses Beispiel nicht benötigt und können gleich gelöscht werden. Dazu zählen zum Erzeugen mehrerer Benachrichtigungen (beispielsweise an mehrere Benachrichtigungsempfänger) die SkripteMultiNotificationProfile.dats und MultiNotificationProfile.xslt. Außerdem ist die NotificationView.xslt zum Formatieren einer freizugebenden Benachrichtigung im P23R-Client optional.
Im Manifest der Benachrichtigungsregel wird als neuer Name addressChangeSend angelegt und der entsprechende Titel eingetragen.
Wie bei der Benachrichtigungsregelgruppe kann festgelegt werden, von wann bis wann eine Benachrichtigungsregel gültig ist, d. h. für welchen Zeitraum eine Benachrichtigung mit dieser Regel generiert werden darf.
Zum Auffinden einer Benachrichtigungsregel soll diese auch mit den wichtigsten Schlagwörtern versehen werden.

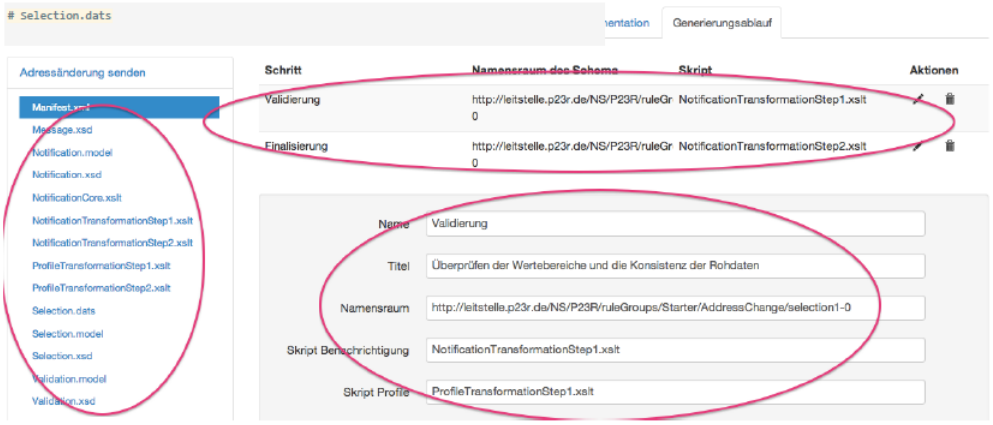


Der Abschnitt "Abgabetermine" bleibt in diesem Beispiel leer, da die Generierung der Benachrichtigung manuell erfolgen soll. In einer verbesserten Version könnte die Generierung der Benachrichtigung regelmäßig angestoßen werden, bspw. wöchentlich entsprechend der Anforderungen, und automatisch überprüft werden, ob sich die Adresse des Unternehmens geändert hat und damit eine entsprechende Meldung rechtzeitig erfolgen muss.
Im Abschnitt "Dokumentation" sollen die Erläuterungstexte, die der Verwalter der zu aktivierenden Benachrichtigungsregelgruppen sieht, entsprechend hilfreich ausgefüllt werden.
Analog sind im Abschnitt "Verantwortlichkeiten" die Ersteller der Benachrichtigungsregel zu vermerken.
Den Kern einer Benachrichtigungsregel bilden die Generierungsschritte, die steuern, in welcher Reihenfolge die verschiedenen für diese Benachrichtigungsregel erstellten Skripte genutzt werden. Die Skripte haben zum Ziel, die Daten im ersten Schritt zu validieren und im zweiten Schritt zu aggregieren und zu transformieren. In diesem

Beispiel werden folgende Schritte mit den angegebenen Ergebnissen (Namespaces) definiert:

- Selection.dats http://.../AddressChange/selected
- NotificationTransformationStep1.xslt http://.../AddressChange/validated
- ProfileTransformationStep1.xslt
- NotificationTransformationStep2.xslt http://.../AddressChange/Notification
- ProfileTransformationStep2.xslt Nachdem die Rohdaten vom Quelldatenkonnektor selektiert wurden, werden diese selektierten Daten im ersten Schritt (Step 1) auf ihre Korrektheit überprüft.

Die validierten Daten werden im zweiten Schritt (Step 2) in das benötigte Format der Benachrichtigung konvertiert. Die Schritte sind entsprechend im Abschnitt "Generierungsschritte" des Manifests einzutragen.



Die Reihenfolge der Einträge im Manifest spiegelt dabei nicht die Reihenfolge der Transformation wider. Diese wird anhand der in jedem Schritt erzeugten Namensräume zur Laufzeit festgelegt.

E.5.7.2. Selektion der Rohdaten

Zunächst werden vor dem ersten Schritt die Daten von den Quelldatenkonnektoren abgefragt. Die Pivot-Teildatenmodelle definieren, in welchem Format die Daten abgefragt werden müssen. Die Abfrage erfolgt durch das P23R-Selection-Skript Selection.dats.

```
# Selection.dats
Zunächst werden die Antragstellerdaten dem Datenmodell Applicant zugeordnet.


    model Applicant = http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/enterpriseCore1-
0


Die Benachrichtigung mit dem Namensraum _Selection_ wird erzeugt.


    selection {
        @xmlns { "http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/selection1-0" }
        [


Die beiden letzten Adressen werden aus dem Unternehmensprofil gefiltert und kopiert.


            for each applicant in Applicant.enterpriseCore.profile
            take 1
            return profile {
                @name {applicant.company@name}


                for each location in applicant.headquarter.locations
                take -2
                return addresses {
                    @validFrom { location@validFrom }
                    @lastModified { location@lastModified }
                    @houseNumber { location.address@housenumber }
                    @street { location.address@street }
                    @city { location.address@city }
                    @zipCode { location.address@zipCode }
                    @country { location.address@country }
                }
            }
        ]
    }
```

Nach dem aktuellen Stand der Technik interpretiert der P23R-Prozessor noch kein DataSel. Daher müssen Anfragen in XQUERY formuliert werden. Ein äquivalenter Ausdruck zum obrigen DataSel ist der folgende XQuery Ausdruck:

```
(: Ziel Namespace der Selektion festlegen :)

declare default element namespace
"http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Selection1-0";

declare copy-namespaces no-preserve, inherit;


(: Deklariere Namespace des NotificationProfils für das XML Fragment :)

declare namespace profile = "http://leitstelle.p23r.de/NS/p23r/nrl/NotificationProfile1-0";

(: Deklariere Namespace der eingehenden Message :)

declare namespace msg =
"http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Message1-0";

(: Deklariere Namespace der p23r-Funktion :)

declare namespace p23r = "http://p23r.de/xquery/datapool";


(: Deklariere Namespaces des NotificationProfils für die p23r-Funktion :)

declare variable $namespace_profile :=
"http://leitstelle.p23r.de/NS/p23r/nrl/NotificationProfile1-0";

(: Deklariere Namespaces des Pivot-Datenmodells :)

declare variable $namespace_piv :=
"http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/enterpriseCore1-0";


(: SELEKTIONS-DEFINITION

Dieses XQUERY Fragment wird an den Quelldaten-Konnektor übergeben.

Nach der Deklaration der Namespaces wird die Selektion definiert.

:)


declare variable $selection := "

    declare namespace pivot =
'http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/enterpriseCore1-0';


    declare namespace selection =
'http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Selection1-0';
```

```
declare variable $sorted-locations :=
        for $location in
/pivot:enterpriseCore/pivot:profile/pivot:headquarter/pivot:addresses
        order by $location/pivot:lastmodified descending
        return $location;

    <selection:profile>
        <company
name='{data(/pivot:enterpriseCore/pivot:profile/pivot:company/pivot:name)}'></company>
        {
        for $location in subsequence($sorted-locations,1,2)
        return
        <addresses
                validFrom='{data($location/pivot:validfrom)}'
                lastModified='{data($location/pivot:lastmodified)}'
                houseNumber='{data($location/pivot:housenumber)}'
                street='{data($location/pivot:street)}'
                city='{data($location/pivot:city)}'
                zipCode='{data($location/pivot:zipCode)}'
                country='{data($location/pivot:country)}'>
            </addresses>
        }
    </selection:profile>
";

(: SELEKTIONS-AUFRUF :)
<selection>
    {p23r:selectDataForNamespace($namespace_piv, $selection)}
</selection>
```

Das Skript Selection.dats liefert als Ergebnis ein XML-Dokument, das dem folgenden Datenmodell Selection entspricht. Das Ergebnis beinhaltet die für den Zweck der Regel notwendigen Inhalte des Pivot-Teildatenmodells, die durch die Selektion ausgewählt und eingeschränkt wurden.

```
model P23R.ruleGroups.Starter.AddressChange.Selection as sel release 1.0 {


    creator "P23R-Team"

    organisation "P23R-Team"

    base http://leitstelle.p23r.de/NS/

    depends on p23r release 1.0


    title "Format für das Ergebnis von Selection.dats"

    description "Vereinfachtes Datenmodell im Vergleich zum Pivot-Datenmodell
enterpriseCore."


    entity Selection {

        profile: Profile optional

    }


    entity Profile {

        company: Company

        addresses: Address many

    }


    entity Company {

        name: string

    }


    entity Address {

        validFrom: date

        lastModified: date

        houseNumber: string

        street: string

        city: string

        zipCode: string

        country: string

    }
}
```

E.5.7.3. Validierung der Rohdaten

Im nachfolgenden ersten Schritt werden die beiden Skripte NotificationTransformationStep1.xslt und ProfileTransformationStep1.xsltausgeführt. Das Skript NotificationTransformationStep1.xslt prüft die Rohdaten mit dem Format des XSD-Schemas Selection.xsd.
(A#) Für die Tests in der Transformation werden zuerst alle Daten aus dem Ast mit

der notification ausgewählt.
(B#) Die Daten der Selektion, die für die Generierung der Benachrichtigung notwendig sind, müssen zunächst als Ergebnis komplett weitergegeben werden, da sie im nächsten Schritt weiterverarbeitet werden sollen.
(C#) Zuletzt folgen die Tests für die selektierten Rohdaten.

```
<?xml version="1.0" encoding="UTF-8"?>


<!-- NotificationTransformationStep1.xslt -->


<xsl:stylesheet
xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Validation1-0"
    xmlns:ng="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationGeneration1-0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:p23r="http://p23r.de/xquery/datapool"

xmlns:selection="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Selecti
on1-0"
    version="2.0"
    exclude-result-prefixes="ng p23r">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>


    <xsl:template match="/">
        <xsl:apply-templates select="/ng:notificationGeneration/ng:notification"/>
    </xsl:template>
```

```
    <!-- A# Auswahl der Daten -->
    <xsl:template match="selection:profile">
        <validation>
            <profile>
                <xsl:apply-templates select="@*|node()" mode="copy"/>
            </profile>
        </validation>
        <xsl:apply-templates select="*" mode="test"/>
    </xsl:template>


    <!-- B# Inhalt der Benachrichtigung kopieren -->
    <xsl:template match="@*|node()" mode="copy">
        <xsl:copy>
            <xsl:apply-templates select="@*|node()" mode="copy"/>
        </xsl:copy>
    </xsl:template>


    <!-- C# Testen -->
    <xsl:template match="selection:profile" mode="test">
        <!-- there must be two addresses -->
        <xsl:if test="count(selection:addresses)!= 2">
            <xsl:value-of select="p23r:protocolWrite('test.fail', 'Es müssen genau zwei
Adressen in der Ergebnismenge sein.')"></xsl:value-of>
        </xsl:if>
        <!-- the validfrom must be different -->
        <xsl:if test="selection:addresses[1]/@validFrom !=
selection:addresses[2]/@validFrom">
            <xsl:value-of select="p23r:protocolWrite('test.fail', 'Die ausgewählten
Adressen müssen unterschiedliche Gültigkeitsdaten haben.')"></xsl:value-of>
        </xsl:if>
    </xsl:template>


</xsl:stylesheet>
```

Das Skript NotificationTransformationStep1.xslt liefert als Ergebnis ein XML-Dokument, das dem folgenden Datenmodell Validationentspricht. Es unterscheidet sich vom vorherigen Datenmodell Selection nur im Namespace und im Wurzelelement. Dabei wird mit rule:/// auf ein bereits definiertes Datenmodell zurückgegriffen. Der Pfad für das Schema wird hierbei relativ zur Benachrichtigungsregel adressChange angegeben.

```
model P23R.ruleGroups.Starter.AddressChange.Selection as sel release 1.0 {

    creator "P23R-Team"
    organisation "P23R-Team"
    base http://leitstelle.p23r.de/NS/
    depends on p23r release 1.0

    title "Format für das Ergebnis von Selection.dats"
    description "Vereinfachtes Datenmodell im Vergleich zum Pivot-Datenmodell
enterpriseCore."

    entity Selection {
        profile: Profile optional
    }

    entity Profile {
        company: Company
        addresses: Address many
    }

    entity Company {
        name: string
    }

    entity Address {
        validFrom: date
        lastModified: date
        houseNumber: string
        street: string
        city: string
        zipCode: string
        country: string
    }
}
```

(A#) Parallel müssen die Metainformationen des NotificationProfile mit dem Skript ProfileTransformationStep1.xslt transformiert werden.
(B#) Da die Metainformationen im Beispiel in diesem Schritt nicht verändert werden sollen, kann der Ast des NotificationProfile ohne Weiteres kopiert werden.
Damit ist der erste Schritt der Generierung beendet.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ProfileTransformationStep1.xslt -->

<xsl:stylesheet
  xmlns="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationProfile1-0"
  xmlns:ng="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationGeneration1-0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

  <xsl:template match="/">
    <xsl:apply-templates select="/ng:notificationGeneration/ng:notificationProfile"/>
  </xsl:template>

  <!-- A# -->
  <xsl:template match="ng:notificationProfile">
    <notificationProfile>
      <xsl:apply-templates select="@*|node()" mode="copy"/>
    </notificationProfile>
  </xsl:template>

  <!-- B# -->
  <xsl:template match="@*|node()" mode="copy">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()" mode="copy"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

E.5.7.4. Erstellung der freizugebenden Benachrichtigung

Im nachfolgenden letzten Schritt werden die beiden Skripte NotificationTransformationStep2.xslt und ProfileTransformationStep2.xsltausgeführt.
Das Skript NotificationTransformationStep2.xslt transformiert die intermediären Daten mit dem Format des XSD-Schemas Validation.xsd in das finale Format des XSD-Schemas Notification.xsd.
(A#) Dazu wird zunächst der Umschlag mit dem Unternehmensnamen erzeugt.
(B#) Anhand des Datums validFrom wird die bisherige und die neue Adresse identifiziert.
(C#) Danach werden beide Adressen im neuen Format der CoreComponents.xsd über-

nommen.

```
<?xml version="1.0" encoding="UTF-8"?>


<!-- NotificationTransformationStep2.xslt -->


<xsl:stylesheet


xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Notification1-0"

    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:ng="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationGeneration1-0"


xmlns:sel="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Selection1-0"


xmlns:val="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Validation1-
0"

    xmlns:cc="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/CoreComponents1-0"
```

```
    version="2.0"
    exclude-result-prefixes="ng">
    <xsl:output indent="yes"></xsl:output>

    <!-- select notification part -->
    <xsl:template match="/">
        <xsl:apply-templates select="/ng:notificationGeneration/ng:notification"/>
    </xsl:template>

    <xsl:template match="ng:notification">
        <xsl:apply-templates select="*" mode="transform"/>
    </xsl:template>

    <!-- Transformation -->
    <xsl:template match="val:profile" mode="transform">

        <!-- store addresses -->
        <xsl:variable name="addressA">
            <xsl:copy-of select="sel:addresses[1]"></xsl:copy-of>
        </xsl:variable>
        <xsl:variable name="addressB">
            <xsl:copy-of select="sel:addresses[2]"></xsl:copy-of>
        </xsl:variable>

        <!-- A# generate envelope and assign current and new address -->
        <notification>
            <companyName><xsl:value-of select="sel:company/@name"></xsl:value-
of></companyName>
            <!-- #B select new and old address-->
            <currentAddress>
                <xsl:choose>
```

```
    version="2.0"
    exclude-result-prefixes="ng">
    <xsl:output indent="yes"></xsl:output>

    <!-- select notification part -->
    <xsl:template match="/">
        <xsl:apply-templates select="/ng:notificationGeneration/ng:notification"/>
    </xsl:template>

    <xsl:template match="ng:notification">
        <xsl:apply-templates select="*" mode="transform"/>
    </xsl:template>

    <!-- Transformation -->
    <xsl:template match="val:profile" mode="transform">

        <!-- store addresses -->
        <xsl:variable name="addressA">
            <xsl:copy-of select="sel:addresses[1]"></xsl:copy-of>
        </xsl:variable>
        <xsl:variable name="addressB">
            <xsl:copy-of select="sel:addresses[2]"></xsl:copy-of>
        </xsl:variable>

        <!-- A# generate envelope and assign current and new address -->
        <notification>
            <companyName><xsl:value-of select="sel:company/@name"></xsl:value-
of></companyName>
            <!-- #B select new and old address-->
            <currentAddress>
                <xsl:choose>
</xsl:stylesheet>
```

(A#) Parallel müssen die Metainformationen des NotificationProfile mit dem Skript ProfileTransformationStep2.xslt transformiert werden.
(B#) Da wir die Benachrichtigung in diesem Beispiel per E-Mail versenden wollen, werden die notwendigen Metainformationen statisch angegeben.
(C#) Der restliche Teil des NotificationProfile kann direkt kopiert werden. Damit ist der letzte Schritt der Generierung der freizugebenden Benachrichtigung beendet.

```xml
<?xml version="1.0" encoding="UTF-8"?>


<!-- ProfileTransformationStep2.xslt -->


<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0"
    xmlns="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationProfile1-0"
    xmlns:ng="http://leitstelle.p23r.de/NS/p23r/nrl/NotificationGeneration1-0">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>


    <!-- A# -->
    <xsl:template match="/">
        <xsl:apply-templates select="/ng:notificationGeneration/ng:notificationProfile"/>
    </xsl:template>


    <!-- C# -->
    <xsl:template match="ng:notificationProfile">
        <notificationProfile>
            <xsl:copy-of select="@*"/>
            <xsl:apply-templates select="node()|comment()"/>
        </notificationProfile>
    </xsl:template>


    <xsl:template match="ng:notificationProfile/*:communication">
        <!-- B# -->
```

```
        <communication>
            <criteria name="dummy"/>
            <publicTimeStamp/>
            <xsl:element name="receivers" namespace="{namespace-uri()}">
                <xsl:attribute name="receiverId" select="umzug"/>
                <xsl:attribute name="receiverName" select="'Amt zur Verwaltung von
Unternehmensadressen'"/>
                <xsl:element name="support" namespace="{namespace-uri()}"/>
                <xsl:element name="technicalSupport" namespace="{namespace-uri()}"/>
                <xsl:element name="channels" namespace="{namespace-uri()}">
                    <xsl:attribute name="id"/>
                    <xsl:attribute name="name">de.fraunhofer.fokus.cc.email</xsl:attribute>
                    <xsl:attribute name="type">system.email.xml</xsl:attribute>
                    <xsl:attribute name="submissionTime">2010-01-
01T00:00:00</xsl:attribute>
                    <xsl:attribute name="timeoutAt">2010-01-01T00:00:00</xsl:attribute>
                    <parameters name="email">umzug@amt.de</parameters>
                </xsl:element>
            </xsl:element>
        </communication>
    </xsl:template>



    <xsl:template match="@*|node()">
        <xsl:copy>
            <xsl:apply-templates select="@*|node()"/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

Das Skript cepFinder.dats muss gelöscht werden, da die Zuständigkeit für die Übermittlung der Benachrichtigung durch den Kommunikationskonnektor statisch im Skript ProfileTransfomration.xslt angegeben wurde.
Die Skripte MultiNotificationProfile.dats und MultiNotificationProfile.xslt können ebenfalls gelöscht werden, da die hier beispielhaft dargestellte Benachrichtigungsregel immer nur einen Benachrichtigungsempfänger hat.
Das Skript representation-Channel Type-Target.xslt kann ebenfalls gelöscht werden, da die freigegebene Benachrichtigung bezüglich ihres Zielformats für die Übertragung im Kommunikationskonnektor nicht mehr angepasst werden muss.
Mit

- dem Ausfüllen des Manifests (insbesondere den Generierungschritten),
- dem Code für Selection.dats,

- NotificationTransformationStep*.xslt,
- ProfileTransformationStep*.xslt,
- den Datenmodellen für Message.xsd,
- Notification.xsd und
- den Zwischenergebnissen ist die Benachrichtigungsregel komplett in einer Minimalform realisiert worden.

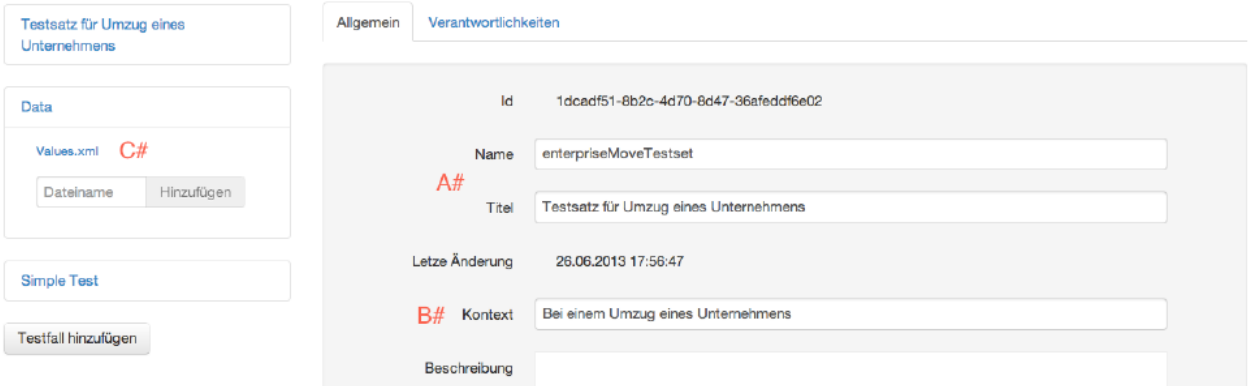
Abschließend muss der Inhalt des Projektes im GIT-Repository gesichert werden, indem unter "Projektinformationen" das Projekt synchronisiert wird.

E.5.8. Qualitätssicherung einer Benachrichtigungsregelgruppe

Nachdem die Pivot-Teildatenmodelle erstellt bzw. angepasst und die Benachrichtigungsregelgruppe implementiert wurden, muss die Benachrichtigungsregelgruppe getestet werden, um sicherzustellen, dass sie auf einem P23R-Prozessor korrekt abläuft. Wenn eine Benachrichtigungsregelgruppe auf der Öffentlichen P23R-Leitstelle veröffentlicht wird, wird diese generell auch auf dem Referenz-P23R-Prozessor dem zugehörigen P23R-Entwicklerportal getestet. Damit ist die Benachrichtigungsregelgruppe auf jedem P23R-Prozessor ablauffähig.
Zum Testen von Benachrichtigungsregeln werden P23R-spezifische Unit-Tests verwendet, wie sie in der T-BRS normativ spezifiziert sind. Diese werden im BDD-Style (Behaviour Driven Development) implementiert.

E.5.8.1. Erstellung eines Testsatzes für eine Benachrichtigungsregelgruppe

Um einen Testsatz mit mehreren Testfällen implementieren zu können, muss wie oben angegeben ein weiteres GIT-Repository und ein Projekt mit der Projektvorlage Testsatz angelegt werden.
(A#) Im Abschnitt "Allgemein" müssen erneut Name und Titel des Testsatzes angegeben werden.
(B#) Im Feld "Kontext" ist anzugeben, worauf sich die einzelnen Testfälle beziehen, beispielsweise wer oder was grundsätzlich im Testsatz getestet wird.
(C#) Zu jedem Testsatz gibt es einheitliche Testdaten.



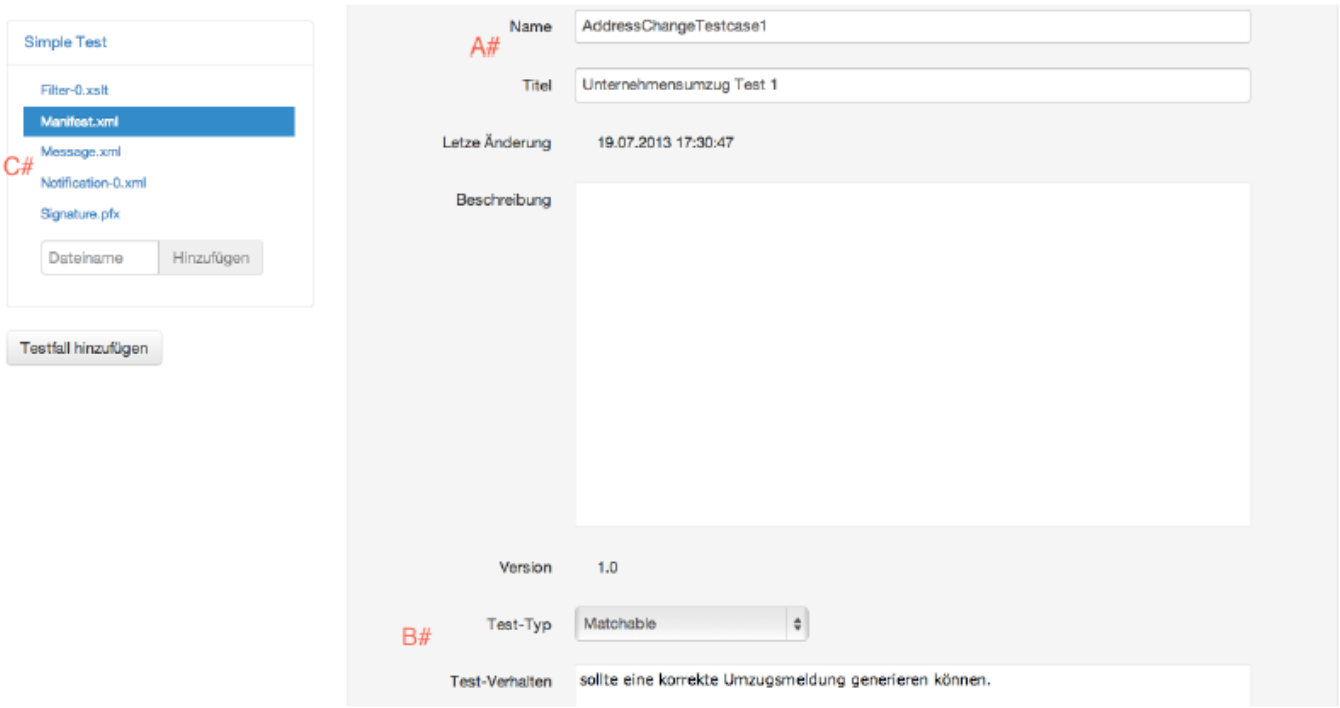
Nachfolgend ist das XML-Dokument mit den Testdaten aufgeführt. Dieses muss alle Daten beinhalten, die im Realbetrieb von den Quelldatenkonnektoren geliefert werden würden. Entsprechend muss der benötigte Namespace vorhanden sein.
Wenn mehrere Quelldatenkonnektoren in der Benachrichtigungsregelgruppe verwendet werden, müssen entsprechend viele XML-Dokumente mit den Testdaten bereitgestellt werden.

```
<?xml version="1.0" encoding="UTF-8"?>

<enterpriseCore
xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/enterpriseCore">

    <profile>

        <company>

            <name>Muster AG</name>

        </company>

        <headquarter>

            <addresses>

              <street>Hubertusallee</street>

              <housenumber>8</housenumber>

              <zipCode>12345</zipCode>

              <city>Berlin</city>

              <country>Germany</country>

              <validfrom>2014-01-01</validfrom>

              <lastmodified>2013-11-01</lastmodified>

            </addresses>

            <addresses>

                <street>Lindenstraße</street>

                <housenumber>16</housenumber>

                <zipCode>12345</zipCode>

                <city>Berlin</city>
```

```
                <country>Germany</country>
                <validfrom>2002-10-02</validfrom>
                <lastmodified>2002-10-02</lastmodified>
            </addresses>
        </headquarter>
    </profile>
    <profile>
        <company>
            <uid>DE080033020</uid>
            <name>Telekom</name>
        </company>
    </profile>
    <profile>
        <company>
            <uid>DE190854732</uid>
            <name>V-Software</name>
        </company>
        <headquarter>
            <addresses>
                <street>Fritschestr.</street>
                <housenumber>30</housenumber>
                <zipCode>2566998</zipCode>
                <city>Berlin</city>
                <country>Brandenburg</country>
                <validfrom>2014-01-01</validfrom>
                <lastmodified>2013-11-01</lastmodified>
            </addresses>
            <addresses>
                <street>Berlinerstr.</street>
                <housenumber>5</housenumber>
                <zipCode>14496</zipCode>
                <city>Berlin</city>
                <country>Brandenburg</country>
                <validfrom>2014-02-01</validfrom>
                <lastmodified>2013-11-02</lastmodified>
            </addresses>
        </headquarter>
    </profile>
</enterpriseCore>
```

Ein Testsatz kann einen oder mehrere Testfälle umfassen.

(A#) Jeder Testfall muss im Abschnitt Allgemein einen Namen und Titel haben.
(B#) Ein Testfall muss entweder funktionieren (matchable) oder scheitern (unmatchable oder error). Entsprechend soll das Verhalten des Testfalls bezogen auf den Kontext beschrieben werden. Kontext und Verhalten werden bei der Ausführung eines Testdurchlaufs protokolliert und erscheinen beispielsweise in der Ergebnisanzeige des P23R-Entwicklerportals.
(C#) Zu jedem Testfall müssen nun die auslösende Nachricht (Message.xml) und das erwartete Ergebnis (Notification_0.xml) implementiert werden.



Nachfolgend ist die auslösende Nachricht (Message.xml) für den Testfall in diesem Beispiel angegeben. Die messageId muss dabei zwangsweise leer sein, damit für jeden Testlauf eine neue, eindeutige ID vom P23R Processor generiert werden kann.

```
<?xml version="1.0" encoding="UTF-8"?>

<message

    namespace="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/Message1-0"

    xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Message1-0"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Mess
age1-0 Message.xsd">
  <messageId></messageId>
  <subject>NeueAdresse</subject>
  <applyAt>2013-11-14</applyAt>
</message>
```

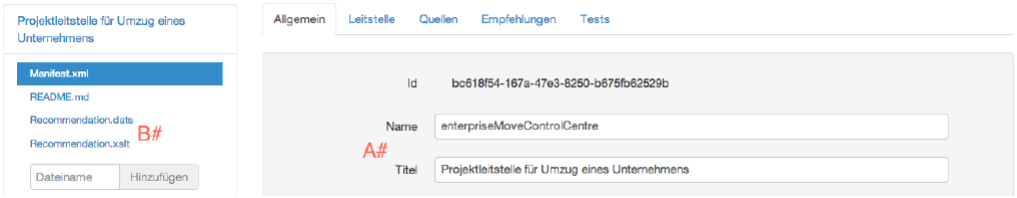
Im Folgenden wird das erwartete Ergebnis (Notification-0.xml) mit der bisherigen und der neuen Adresse des Unternehmens für den Testfall dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:cc="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/CoreComponents1-0"

xmlns="http://leitstelle.p23r.de/NS/P23R/ruleGroups/Starter/AddressChange/Notification1-0">
    <companyName>Muster AG</companyName>
    <currentAddress>
        <cc:identification>2002-10-02</cc:identification>
        <cc:streetName>Lindenstraße</cc:streetName>
        <cc:buildingNumber>16</cc:buildingNumber>
        <cc:cityName>Berlin</cc:cityName>
        <cc:postcode>12345</cc:postcode>
        <cc:country>Germany</cc:country>
    </currentAddress>
    <newAddress>
        <cc:identification>2014-01-01</cc:identification>
        <cc:streetName>Hubertusallee</cc:streetName>
        <cc:buildingNumber>8</cc:buildingNumber>
        <cc:cityName>Berlin</cc:cityName>
        <cc:postcode>12345</cc:postcode>
        <cc:country>Germany</cc:country>
    </newAddress>
</notification>
```

Das Skript Filter-0.xslt kann ebenfalls gelöscht werden, da das Ergebnis eines Testfalls direkt mit dem erwarteten Ergebnis verglichen wird und keine Daten gefiltert werden.
Die Signatur Signatur.pfx kann ebenso gelöscht werden, da die freizugebende Benachrichtigung in diesem Beispiel nicht signiert wird.
Abschließend muss der Inhalt des Projektes im GIT-Repository gesichert werden, indem unter "Projektinformationen" das Projektsynchronisiert wird.

E.5.8.2. Veröffentlichung in der Projekt-Leitstelle

Um Benachrichtigungsregelgruppen testen zu können, müssen sie zuerst in einer Projekt-Leitstelle, sprich deren P23R-Depot, veröffentlicht werden. Diese P23R-Depots sind passwortgeschützt, können aber von jedem P23R-Prozessor genutzt werden. Insbesondere verwendet der im P23R-Entwicklerportal integrierte Test-P23R-Prozessor dieses P23R-Depot für Testläufe.
Leitstellen sollen von nur einem Verantwortlichen bearbeitet werden. Das P23R-Entwicklerportal unterstützt nicht die Bereitstellung eines Projektleitstellen-Projektes über mehrere Benutzer-Konten.
Zuerst muss nochmals ein GIT-Repository und ein Projekt mit der Projektvorlage Projekt-Leitstelle angelegt werden.
(A#) Im Manifest im Abschnitt "Allgemein" müssen Name und Titel der Projekt-Leitstelle eingetragen werden.
(B#) Die Dateien Recomendation.dats und Recomendation.xslt müssen nicht verändert werden, da das in der P23R-Leitstelle automatisch generierte Benachrichtigungsregelpaket immer empfohlen werden soll.
Im Abschnitt "Leitstelle" muss der Nutzername und das Passwort eingetragen werden, das für den Zugriff auf die Projekt-Leitstelle verwendet werden muss.



(A#) Im Abschnitt "Quellen" müssen alle GIT-Repositories aufgeführt werden, deren Inhalte in dem Benachrichtigungsregelpaket und dem Datenmodellpaket im P23R-Depot der Projekt-Leitstelle veröffentlicht werden sollen.

Unter "Empfehlungen" werden die Namespaces der Quelldatenkonnektoren angegeben, die das Recommendation.dats Skript benötigt, um zu ermitteln, ob das Benachrichtigungsregelpaket dem Anwender empfohlen werden soll.
Nachdem das Manifest mit Aktualisieren gespeichert wurde, müssen die Quellen mit Quellen aktualisieren geladen werden.
Nun können im Abschnitt Tests (A#) Test-Pakete ausgewählt und (B#) Testläufe gestartet werden. Es wird angezeigt, ob alle Testfälle fehlerfrei durchlaufen (C#).

Abschließend muss der Inhalt des Projektes im GIT-Repository gesichert werden, indem wir unter "Projektinformationen" das Projekt synchronisieren.
Damit ist die Entwicklung des Beispiels abgeschlossen. Es hat alle Schritte gezeigt, die auch umfangreichere Benachrichtigungsregeln und -regelgruppen benötigen.

E.5.9. Und wie geht es weiter?

Nachdem die Entwicklung und Implementierung von Benachrichtigungsregelgruppen grundsätzlich verstanden worden sind, empfiehlt es sich, die Spezifikation der T-BRS nochmals genauer durchzulesen. Ferner sind unter Info/FAQ einige Detaillösungen für typische Probleme bei der Erstellung von Benachrichtigungsregeln aufgeführt.

F. Konzept für die Architektur und Funktionalität eines P23R4FLEX-Clients

Das folgende Kapitel ist dem Abschlussbericht der ENDA KG entnommen.
Das P23R-Prinzip sieht derzeit für die Datenlieferung („Benachrichtigungs-Auslieferung“) entweder Individuallösungen oder einen P23R-Client vor. Einen P23R-Client, der auch Daten liefern kann, gibt es derzeit nicht. Er müsste nach der Spezifikation sowohl die neue Abfragesprache DataSel beherrschen, selbstständig auf in unterschiedlicher Form vorliegende Unternehmensdaten zugreifen und falls Daten fehlen, autonom andere P23R-Clients nach den fehlenden Daten fragen können. Individuallösungen sind hinderlich, da teuer, wenn das P23R-Prinzip bei einer Vielzahl von Unternehmen oder anderen P23R-Datenlieferanten zum Einsatz kommen soll. Es wird eine Alternative gesucht und beschrieben, die den Anforderungen im Umweltbereich gerecht werden kann und mit insgesamt geringerem Entwicklungsaufwand verbunden ist.

F.1. Vorgehensweise

Nachfolgend wird eine Analyse typischer Berichtsprozesse im Umweltbereich vorgenommen. Ziel ist dabei, wiederkehrende Aufgaben zu identifizieren und zu beschreiben. Dabei werden nur solche Aufgaben betrachtet, die automatisierbar sind. Neben den für den Umweltbereich charakteristischen Randbedingungen sind die Effekte des P23R-Prinzips zu berücksichtigen. Hier wird das Augenmerk auf die Nutzung eines P23R-Clients durch KMU gerichtet. Für die Aufgaben werden Lösungen skizziert und Randbedingungen einer Automatisierungslösung unter Nutzung des P23R-Prinzips formuliert. Sind mehrere verschiedene Lösungsansätze für eine Aufgabe erkennbar, werden deren Vor- und Nachteile diskutiert, um eine Implementierungsempfehlung geben zu können. Auf Basis der Lösungsvorschläge wird abschließend eine grobe Aufwandsabschätzung vorgenommen.

F.2. Aufgaben innerhalb der Umweltberichterstattung

Umweltberichterstattung wird grundsätzlich auf der Basis bestehender Rechtserkenntnisquellen durchgeführt. Es sind in jedem Fall kommunale, Landes- und/oder Bundesbehörden mit dem Vollzug befasst. Die Datenquellen liegen sowohl bei Unternehmen oder nichtstaatlichen Organisationen als auch bei staatlichen Organisationen oder Behörden. Einige Berichtspflichten werden nach festen Zyklen fällig und sind zu festen Stichtagen zu erfüllen, einige sind auf Anforderung innerhalb einer Frist zu bedienen.
Der strukturelle Datenumfang ist in den jeweiligen Rechtserkenntnisquellen bisher nur in wenigen Fällen so genau definiert, dass eine Interpretation zur Festlegung

des tatsächlichen Datenumfangs unterbleiben könnte. Der quantitative Datenumfang hingegen ist in aller Regel hinreichend genau spezifiziert.
Die Datenübermittlung erfolgt papiergebunden oder in einer Tabellenkalkulation, wenn sie schlecht organisiert ist, oder aber in wohldefinierten elektronischen Formaten, wenn sie gut organisiert ist.
Ein großer Teil der entstehenden Aufwände bei der Berichterstattung von Umweltdaten entfällt auf die Genese der einzelnen Werte, auf deren Zusammenstellung und auf die Überführung in eine elektronische Form. Im günstigsten Fall handelt es sich um Labormesswerte, die von einem Messgerät bereits elektronisch in ein Laborinformationssystem übertragen werden und die mit einem Probencode versehen sind, der es ermöglicht, den Messwert automatisiert in einem Fachsystem in genau dem richtigen fachlichen Kontext abzulegen. In der Regel werden auch Prüfungen der zu berichtenden Daten durchgeführt.
Die Genese der einzelnen Werte wird durch P23R nicht berührt, ist nicht nur im Umweltbereich, sondern generell notwendig und wird daher hier nicht weiter diskutiert. Auch die Zusammenstellung der Daten und ihre Überführung in eine elektronisch lesbare Form sind Voraussetzung für eine weitere Verarbeitung oder Übermittlung durch jedwedes elektronische System. Elektronisch lesbare Formen zeichnen sich dadurch aus, dass mit vertretbarem Aufwand ein Algorithmus entwickelt und programmiert werden kann, der die so vorliegenden Informationen nicht nur als Ganzes lesen, sondern auch in Teilbereiche mit bekannter Bedeutung zerlegen kann. Eine Word-Datei ist zwar meist elektronisch lesbar, verfügt aber in den wenigsten Fällen über die Struktur, um automatisch und inhaltlich korrekt zerlegt zu werden. Dies trifft in aller Regel auch auf Excel-Dateien zu.
Umweltdaten enthalten häufiger als andere Datenmodelle Listenattribute. Das Datenmodell der elektronischen Bodendatenbank eBIS enthält über 200 Listen bei insgesamt ca. 350 Attributen. Dies sind Attribute, deren Wert einer Liste (einem Katalog) entstammt. Textfelder sind in guten Umweltdatenmodellen weniger häufig, da sie nur manuell sinnvoll auswertbar sind. Weitere Eigenschaften von Umweltdatenmodellen sind ihre recht hohe Zahl von Attributen (strukturelle Breite) sowie die vergleichsweise hohe Zahl an Qualitätsprüfungen, die für Umweltdaten gefunden werden kann.

F.3. P23R-spezifische Randbedingungen der Berichterstattung

Eine breite Nutzung des P23R-Prinzips setzt voraus, dass auch KMUs den P23R nutzen. Diese verfügen nicht über die finanziellen Möglichkeiten großer Unternehmen und sind aufgrund des geringeren Sparpotenzials auch in geringerem Maße zu Investitionen in die Entwicklung eines eigenen P23R-Clients bereit. Folglich muss für KMUs eine preiswerte Lösung gefunden werden.
Die spezifischen Funktionen eines P23R-Clients sind:

- Auslösen einer P23R-Regel, die die zu berichtenden Daten von einem P23RSourceConnector erhält und verarbeitet,
- Empfangen einer Meldung, dass eine Benachrichtigung erstellt wurde,
- Abholen der verarbeiteten Daten vom P23R-Prozessor,

- Darstellung der zu berichtenden Daten (wenn die P23R-Regel diese erfolgreich verarbeitet hat),
- Darstellung einer Fehlermeldung, die von der P23R-Regel generiert wurde,
- Optional: Umwandlung der zu berichtenden Daten in ein für den Anwender leicht lesbares/druckbares Format, z. B. PDF, und Möglichkeit der Speicherung als Datei,
- Möglichkeit der Freigabe zu berichtender Daten beim P23R-Prozessor,
- Optional: Darstellung einer Rückmeldung des P23R-Prozessors auf die Freigabe.

Die Funktionen, die mit Abstand den größten Implementationsaufwand mit sich ziehen, sind die Darstellungsfunktionen und die Umwandlung in ein leicht lesbares Format. Diese Funktionen sind gleichzeitig stark abhängig von der Art der Daten und müssen für jede P23R-Regel individuell programmiert werden.
Änderungen, deren Notwendigkeit bei der Betrachtung der zu berichtenden Daten offenbar werden, sollten in jedem Fall in den Quellsystemen erfolgen. Daher ist die Speicherung der zu berichtenden Daten durch den P23R-Client nicht notwendig und auch nicht sinnvoll.

F.3.1. Empfangen der Meldung „Benachrichtigung wurde erstellt“

Der Empfang der Meldung „Benachrichtigung wurde erstellt“ wird hier gesondert diskutiert, da beim Betrieb eines P23R-Prozessors in einer externen (Cloud-) Umgebung der Prozessor ggf. von außen auf einen internen P23R-Client des Datenlieferanten zugreifen können muss.
Konkret ist dies ein SOAP-Service, der folgende Funktionalität hat: Empfangen einer Meldung („IApprovalNotify“) des P23R-Prozessors, dass eine Benachrichtigung erstellt wurde und dass diese unter Angabe einer bestimmten ID vom P23RProzessor bezogen werden kann. Nach Empfang dieser Meldung muss der Client die Benachrichtigung vom P23R-Prozessor abrufen und ggf. lokal speichern, um eine erneute Anforderung zu vermeiden.
Implikation: Der Client muss dauerhaft laufen, um die Meldung empfangen zu können, dass eine Benachrichtigung erstellt wurde. Alternativ könnte ein dedizierter Service implementiert werden, der diese Aufgabe erfüllt.

F.4. Lösungsvorschläge für funktionale Anforderungen

Die P23R-spezifischen Funktionen, die eine sehr geringe Abhängigkeit von der Art der zu berichtenden Daten aufweisen, können in nahezu unveränderter Form für die unterschiedlichsten Berichte eingesetzt werden. Dies sind:

- Auslösen einer P23R-Regel, die die zu berichtenden Daten von einem P23RSourceConnector erhält und verarbeitet,
- Warten auf die Meldung, dass die Benachrichtigung bereit liegt,
- Abholen der verarbeiteten Daten vom P23R-Prozessor,

- Möglichkeit der Freigabe zu berichtender Daten beim P23R-Prozessor,
- Optional: Darstellung einer Rückmeldung des P23R-Prozessors.

Bei diesen Funktionen ist es geboten, die veränderlichen Parameter (z. B. „Welche Regel soll auf welchem P23R-Prozessor ausgelöst werden?“) in Form von Konfigurationsdaten zu hinterlegen. Somit kann eine Anpassung des Quellcodes unterbleiben und es werden stattdessen Konfigurationsdateien angepasst. Werden diese Funktionen fertig nutzbar angeboten, muss sich ein P23R-Client-Entwickler nicht mit den besonderen Spezifikationen und Funktionsweisen des P23R-Prinzips auseinander setzen. Die aufwändiger zu implementierenden Funktionen

- Darstellung der zu berichtenden Daten (wenn die P23R-Regel erfolgreich verarbeitet hat),
- Darstellung einer Fehlermeldung, die von der P23R-Regel generiert wurde und die
- optionale Umwandlung der zu berichtenden Daten in ein für den Anwender leicht lesbares/druckbares Format, z. B. PDF, und Möglichkeit der Speicherung als Datei sind bezüglich der Programmierung hingegen völlig frei von P23R-Spezifika.

F.5. Umweltspezifische Randbedingungen der Berichterstattung

Die umweltberichtsspezifischen Anforderungen ergeben sich aus den genannten Randbedingungen der vergleichsweise großen strukturellen Breite sowie der großen Zahl an möglichen Qualitätsprüfungen.
Dies wirkt sich zunächst auf die Darstellung der zu berichtenden Daten aus.
Es können dem Anwender keine XML-Inhalte präsentiert werden, sondern es wird eine nach Umfang der darzustellenden Berichtsentitäten unterteilte Formulardarstellung der Inhalte benötigt.
Zumindest bei umfangreichen Datenlieferungen, wie sie bei größeren Unternehmen der Fall ist, wird für die effiziente Bearbeitung eine Datensatznavigation benötigt. Dies könnte mit einer kompakten Liste realisiert werden, aus der der Anwender stichprobenartig einzelne Objekte zur Formulardarstellung für die Detailprüfung auswählen kann.
Auch die optionale Umwandlung der zu berichtenden Daten in ein für den Anwender leicht lesbares/druckbares Format schließt die Ausgabe von XML-Daten aus. Hier wird eine strukturierte und gut lesbare Darstellung benötigt, deren Zielstellung jedoch nicht die Sichtprüfung der Datensätze oder deren Publikation ist, sondern die Ablage in einer Form, die bei der Klärung von Sachverhalten zwischen Berichter und Empfänger als Basis verwendet werden kann. Dies stellt deutlich geringere Anforderungen an die optische Gefälligkeit des Layouts und etwas höhere an die eindeutige Identifizierbarkeit der Zugehörigkeit von Werten zu bestimmten Objekten.

F.6. Lösungsvorschläge für die konzeptionelle Umsetzung

Das P23R-Prinzip betont in seinen Anforderungen die Nutzbarkeit durch kleine und mittlere Unternehmen (KMU). Während dies nur geringe Auswirkungen auf die Betrachtung der Funktionalität hat, ist es bei der Betrachtung von Umsetzungskonzepten von weitreichender Bedeutung.
Dies ist dadurch begründet, dass Großunternehmen in nur begrenztem Maß auf vorhandene oder vorgefertigte Lösungen angewiesen sind, da ihnen mehr Kapital und somit Mittel für die Programmierung individueller Lösungen zur Verfügung stehen. KMU benötigen weitgehend vorgefertigte oder komplett einsatzbereite Systeme, die ihnen die Einarbeitungszeit in die P23R-Schnittstellenstruktur sparen und die Entwicklungskosten senken. Im Gegenzug nehmen sie eingeschränkte Gestaltungsmöglichkeiten in Kauf. Dieser Weg steht natürlich auch den Großunternehmen offen.

F.6.1. Differenzierung konzeptioneller Schwerpunkte

Aus Sicht eines Großunternehmens mit etablierten IT-Anwendungen für die unternehmensinterne Auswertung von Umweltdaten stehen Mehrbenutzerfähigkeit und Rollen- Rechtekonzepte sowie möglichst nahtlose Integration der P23R-Clientfunktionalität in die unternehmensinternen Prozesse und die Oberflächen der bestehenden IT-Anwendungen im Vordergrund.
Kleine und mittlere Unternehmen brauchen möglichst kurzfristig einsetzbare Lösungen, die ohne umfangreiche Entwicklungsarbeiten mit den notwendigen Berichtsdaten versorgt werden können und in der bestehenden Unternehmensumgebung, ggfs. auf einem Mitarbeiter-PC, in Betrieb genommen werden können. Eine nahtlose Integration in bestehende Systeme ist nicht prioritär, insbesondere dann nicht, wenn die Nutzungsoberfläche robust und selbsterklärend ist.
Im Fall einer Webanwendung ist ein Hosting im Unternehmen für KMU nicht zwingend.

F.6.2. P23R-Client: Was ist möglich, was ist sinnvoll?

Fat-Client-Anwendungen, also klassische, lokal installierte Programme sind aus der Mode gekommen. In größeren Organisationen werden sie inzwischen häufig über Terminalserver bereit gestellt, also zentral installiert und dezentral angezeigt. Dies ist dem Arbeitsprinzip von Webanwendungen ähnlich und erlaubt es, die IT-Dienstleistungen an zentralen Orten zu bündeln und die für den Vor-Ort-Support vorgehaltenen Ressourcen zu reduzieren.
Ohne Vor- und Nachteile von FAT-Client- und Web-Anwendungen detailliert gegeneinander abzuwägen, muss konstatiert werden, dass die Akzeptanz von Webanwendungen heute deutlich höher ist und diese auch als Stand der Technik gelten. Es wird folglich empfohlen, P23R-Clients als Webanwendungen zu entwickeln. Eine P23R-Client-Webanwendung als Lösung, die ohne weitere Zuarbeiten für alle Regeln funktioniert, kann es nicht geben. Dies ist dem Umstand geschuldet, dass die zu berichtenden Daten jeweils unterschiedliche Nachrichtenmodelle mit unterschiedlichen Namen der Attribute und Entitäten besitzen.
Ebenso ist es nicht wirtschatflich möglich, die jeweils (für eine Regel, also eine Berichtspflicht) notwendigen Berichtsdaten automatisiert aus beliebigen IT-Systemen

zu extrahieren, ohne dafür Programmcode zu schreiben. Dies mag zwar theoretisch möglich sein, aber der Aufwand zur strukturierten und umfassenden Beschreibung aller vorliegenden Informationen als perfekte Metainformation übersteigt den Aufwand für ein spezifisches Extraktionsprogramm (also einen SourceConnector) bei weitem.
Seit Anbeginn der IT verbreitet und erfolgreich eingesetzt ist hingegen die Wiederverwendung von Programmcode in Form von dokumentierten Funktionssammlungen in Bibliotheken, heute gerne als Toolkits bezeichnet.

Generelle Vorgaben

Es wird ein Toolkit zur Erstellung eines individuellen P23R-Clients benötigt. Dieses Toolkit muss begleitet werden von einem im Betrieb befindlichen Anwendungsbeispiel, das als Vorlage dient.
Das Toolkit darf keine Musterimplementierung sein, die nur in einem einzigen Fall unter Laborbedingungen funktioniert, sondern muss auf den Produktiveinsatz zugeschnitten und robust sein. Die verwendeten Werkzeuge müssen offen und frei sein, da werden beim Einsatz der Werkzeuge noch der damit erstellten Produkte Lizenzkosten entstehen dürfen.
Die Online-Dokumentation sollte in einem kollaborativen Werkzeug abgebildet sein, das es den Nutzern erlaubt, Anmerkungen, Erweiterungen und Korrekturen vorzunehmen. Als Nutzer kann sich jedefrau/jedermann anmelden.

Geeignet für den Produktiveinsatz

Das Toolkit muss einfach zu benutzen sein. Dies bedeutet nicht, dass es Unerfahrene sofort und fehlerfrei einsetzen können müssen. Es sollte jedoch Entwicklern ermöglichen, konkrete Ergebnisse zu erreichen, ohne zuvor viele Seiten fachlicher Spezifikationen durcharbeiten zu müssen. Es sollte als eine strukturierte, durchsuchbare und stilistisch möglichst homogene Online-Dokumentation verfügbar sein, die als Community-Portal frei zur Verfügung steht und von Nutzern kommentiert, korrigiert und erweitert werden kann.

Robustheit

Robustheit bedeutet, dass ein möglichst breites Spektrum an fehlerhafter Verwendung mit entsprechenden Fehlermeldungen behandelt werden muss, um den Entwicklern die Möglichkeit zu geben, ihre Fehler zu korrigieren. Kommentarlose Abstürze, die keinerlei Hinweise auf ihre Ursache geben, sollten möglichst vermieden werden.

Einsatz möglichst etablierter Technologien

Als Entwicklungssprache sollte Java eingesetzt werden, denn die Zahl der Entwickler, die einen P23R-Client mit Hilfe des P23R Client-Toolkit entwickeln können, soll möglichst groß sein.

Als Bibliotheksbasis wird der Einsatz derjenigen Java Enterprise Edition (JEE)-Funktionen empfohlen, die auch als .jar-File zur Ergänzung von Servlet-Containern zur Verfügung steht. Damit ist gewährleistet, dass der P23R-Client auf unterschiedlichen Servern (Wild- Fly (ehem. JBoss), Glassfish, Oracle WebLogic, IBM WebSphere, Resin, Apache Tomcat) genutzt werden kann.
Die Zahl der Java Frameworks für Rich Internet Applications (RIA) ist sehr groß und ein konkretes Votum führt unweigerlich zum Widerspruch all jener, die das entsprechende Framework nicht einsetzen.
Da jedoch ohnehin ein Toolkit mit funktionierendem Anwendungsbeispiel bereit gestellt werden soll, ist umfangreiche Erfahrung mit dem Web-Framework nicht Voraussetzung für dessen Einsatz und es werden hier allgemeine Anforderungen formuliert:
Eine moderne Lösung sollte auf serverseitiger Architektur in Kombination mit Ajax basieren. Der Einsatz von Browser-Plugins ist aufgrund des vielfältigen Nutzer- und Browserspektrums nicht sinnvoll. Ist SAGA-5-Konformität unabdingbar, muss mit Ajax und Java- Script auf moderne Lösungen und die weitaus meisten Frameworks verzichtet werden. Dieser Fall wird hier nicht weiter diskutiert.
Das Framework sollte seit mindestens drei Jahren etabliert sein und Sprünge in der Major Version mit weitreichenden Änderungen der Schnittstellenphilosophie sollten maximal alle zwei Jahre, besser seltener auftreten. Die Ausrichtung der Auswahl sollte auf ein vernünftiges Verhältnis von Langlebigkeit zu einfacher Handhabung und Funktionsumfang mit Schwerpunkt auf Langlebigkeit ausgerichtet sein.
Der Einsatz spezifischer Bibliotheken, die nur in bestimmten, schlimmstenfalls älteren Versionen mit bestimmten anderen Versionen eines JEE-compliant Webservers zusammenarbeiten, ist unbedingt zu vermeiden. Negativbeispiele sind der Einsatz von Switchyard, der mit keiner anderen Version als JBoss EAP Version 6.1 GA, geschweige denn mit Apache Tomcat oder Resin oder Glassfish arbeitet oder das JBoss Seam Framework, das beim Sprung von Version 2 auf 3 wichtige Funktionen nicht mehr enthielt, dessen Version 2 jedoch jahrelang nicht mit aktuellen JBoss-Versionen zusammenarbeitete.
Im Zweifelsfall sollte eine einfache Funktion selbst implementiert werden, anstatt ein weiteres Framework einzubinden und sich damit eine weitere Versionsabhängigkeit einzuhandeln.

User Interface-Design

Das User Interface-(UI-) Design sollte den Bedienprozess unterstützen, einfach und übersichtlich sein. Ein prototypischer Bedienprozess (kein Beispiel für eine ergonomische Benutzungsoberfläche!) für den P23R-Client ist in Abb. F.1 dargestellt.

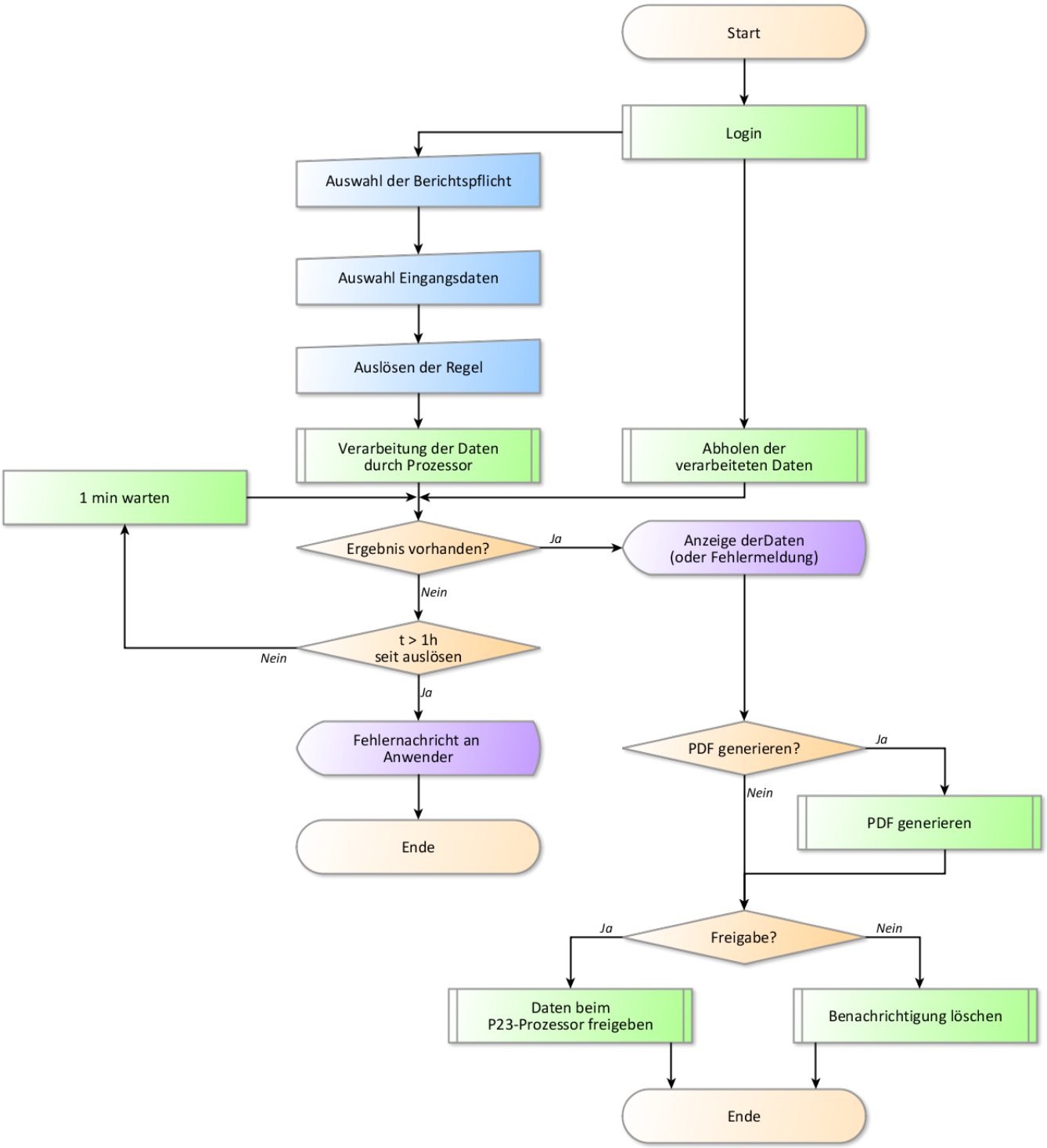


Abbildung F.1.: Workflow-Diagramm „Umweltberichterstattung mit P23R-Client“

Ease of use

Die spezifischen Funktionen (Kapitel F3) des P23R-Clients geben zusammen mit dem Workflow-Diagramm bereits einen klaren Rahmen vor.
Augenmerk sollte auf die clientseitige Benutzerführung nach dem Auslösen der Regel gelegt werden. Wird der SOAP-Service zum Empfang der Meldung „Benachrichtigung wurde erstellt“ vom P23R-Client separiert, um den Zugriff von außen in die interne IT-Umgebung des Datenlieferanten zu unterbinden, besteht ohne eine separate Anfrage keine Möglichkeit, herauszufinden, ob die Regel bearbeitet wurde. Es bestehen dann folgende Möglichkeiten, den Workflow wieder aufzunehmen:

1. Der Anwender wird gebeten, in einigen Minuten nachzusehen, ob die Regel die Benachrichtigung fertig erstellt hat. Dazu bietet das User Interface einen Knopf, den der Anwender sofort als dafür zuständig identifizieren kann. Dies ist nicht sehr komfortabel, kann jedoch SAGA 5-konform umgesetzt werden.

2. Die Webanwendung setzt AJAX ein, um dem Anwender eine „in Bearbeitung“-Meldung zu zeigen und parallel in zeitlichen Abständen den P23R-Prozessor abzufragen, ob die Regelbearbeitung abgeschlossen wurde. Dies kann wegen des notwendigen Einsatzes von JavaScript zwar nicht SAGA 5-konform erfolgen, ist aber das übliche Vorgehen. Dieses Vorgehen wäre auch ohne den o.g. separierten Service notwendig, wenn der Anwender mehr Informationen erhalten soll, als dass die Seite lädt (oder „hängt“).

Im Sinne einer einfachen und klaren Benutzerführung ist Möglichkeit 2 zu bevorzugen.

Realisierung der Stylesheets

Für das Design kann auf das offene und freie, im Rahmen des MIFLEX-Projekts von der FH Potsdam entwickelte, Design9 zurück gegriffen werden.
Die für das P23R-Client-Toolkit zu entwickelnden Stylesheets sollten auf Wiederverwendung ausgelegt sein und verständliche ClassNames und unique IDs (für die Referenzierung der einzelnen Selektoren) enthalten.
Für die Darstellung der Berichtsdaten sind Navigation und Objektlisten nur für solche Berichterstattungen notwendig, bei denen verzweigte Strukturen und ein größerer Umfang zu überprüfender Berichtsobjekte auftritt.
Für die Entwicklung eines P23R-Client-Toolkits sollte dies berücksichtigt werden, so dass Navigation und Objektlisten sowohl leicht integriert als auch weggelassen werden können.

PDF-Generierung

Für die Generierung von PDF-Dateien stehen verschiedene freie Werkzeuge (z. B. XSLFO, ggf. auch Apache PDFBox), zur Verfügung. Sollen Microsoft-Dokumente (Word, Excel etc.) generiert werden, kann Apache POI eingesetzt werden; Mit JOpenDocument können PDF- und OpenDocument-Dateien erstellt werden.
XSL-FO ist als Werkzeug zur Erstellung PDF-Dateien seit Jahren etabliert, stabil funktionsfähig und gut dokumentiert, weshalb dessen Einsatz hier empfohlen wird.

F.7. Abschätzung der Realisierungsaufwände

Die hier vorgenommene Abschätzung der Realisiserungsaufwände kann nur eine grobe Näherung sein, da diese von zahlreichen nicht-fachlichen Faktoren, der individuellen Ausgestaltung einer Ausschreibung sowie ggfs. hier nicht berücksichtigten Arbeiten beeinflusst wird.
Im folgenden Text sind die derzeit absehbaren Arbeiten aufgeführt. Die Aufwandsschätzungen sind in tausend Euro brutto (k€) kursiv angegeben:

1. Implementation der Kommunikationsfunktionen zwischen P23R-Client und P23RProzessor (Aufwand ca. 25k€)

   - Regelverarbeitung beim Prozessor auslösen
   - Empfangen der Meldung „Benachrichtigung wurde erstellt“ (Inklusive einem dezidierten Service, der zum Einsatz kommt, wenn keine externen Zugriffe auf die interne IT des Datenlieferanten gewünscht sind.)
   - Verarbeitungsergebnis beim Prozessor abholen
   - Ergebnis freigeben
   - Ergebnis löschen (keine Freigabe)
   - Aktualität einer Regel beim Prozessor erfragen (Diese Funktionalität wird im Mai 2014 vom FOKUS P23R-Prozessor V3.2 und V4.0 noch nicht unterstützt.)
   - Regel aktualisieren

2. Entwicklung eines Stylesheets für die Login-Maske
3. Implementation von Login und Authentifikation
4. Entwicklung eines Stylesheets für die Haupt-Maske
5. Implementation der Haupt-Maske

   - UI (User Interface) Regel/Berichtspflicht auswählen
   - UI Regelverarbeitung beim Prozessor auslösen
   - UI und Ajax-Funktion für automatisierte Anfrage des Verarbeitungsergebnisses beim Prozessor
   - UI Verarbeitungsergebnis beim Prozessor manuell anfragen. Falls der Anwender das Browserfenster mit der automatisierten Anfrage schließt, muss die Möglichkeit bestehen, das Ergebnis später (z. B. am nächsten Tag) zur Prüfung abzuholen.
   - UI PDF-Export der Berichtsdaten
   - UI Ergebnis freigeben
   - UI Ergebnis löschen (keine Freigabe)

- UI Aktualität einer Regel beim Prozessor erfragen
- UI Regel aktualisieren

Mit Stylesheets und Login zusammen (Aufwand ca. 20k€)

6. UI: Implementation der Darstellung der freizugebenden Berichtsdaten. Dieser Teil ist in hohem Maß abhängig von der Berichtspflicht.

- Entwicklung eines Stylesheets für die Objekt-Tabelle (mindestens wechselnde Zeilenhintergründe zur Erleichterung des Lesens)
- UI für die Darstellung der Berichtsobjekte als Zeilen einer Tabelle mit jeweils einem oder mehreren Links auf Formulardarstellungen der Berichtsobjektdetails.
- UI für Pagination (Navigation zwischen portionierten Tabellenausschnitten, Zeilenzahl konfigurierbar)
- UI für die Navigation innerhalb eines Berichtsobjekts, wenn dieses aus mehreren, ggf. umfangreichen Entitäten besteht und zur Rückkehr zur Hauptseite oder Tabellenübersicht.
- Entwicklung eines Stylesheets für ein Standard-Objekt-Formular mit unterschiedlichen Klassen für unterschiedliche Attributtypen (Zahlen, Kurztexte, Langtexte, Listenwerte, Boolesche Werte)
- UI für die Darstellung eines Berichtsobjekts als Formular mit Label/Entry- Feldern (Entries gesperrt) mit Klassen zur Nutzung der Stylesheet- Darstellungen für unterschiedliche Attributtypen.

(Aufwand ca. 30k€)

7. Implementation einer Funktion zur Erstellung einer PDF-Datei mit den Berichtsinhalten als Kopiervorlage bzw. Beispiel. (Diese Funktion ist wie auch die Darstellungsfunktion von den Berichtsinhalten abhängig und muss jeweils individuell entwickelt werden.)

(Aufwand ca. 15k€)

8. Entwickeln eines Standard-Nutzungsszenarios für das zu entwickelnde P23R-Client- Toolkit ohne Notwendigkeit der Darstellung einer Objektübersicht und Navigation zwischen verschiedenen Berichtsformularen. Szenario heißt, folgendes zu dokumentieren:

- welche Aufgaben in welcher Reihenfolge ein Entwickler jeweils angehen sollte/ angehen wollen wird
- auf welche Artefakte (z. B. XML-Beispieldateien oder einen Demo-Prozessor) der Entwickler bei der jeweiligen Aufgabe zurückgreifen muss
- welche Fragen sich der Entwickler dabei stellen könnte oder welchen Schwierigkeiten er sich ausgesetzt sehen könnte (■ Lösungsvorschläge)

9. Entwickeln eines Erweiterungs-Szenarios mit Objektübersicht, Pagination und verschiedenen Formularen für die Entitäten eines Berichtsobjekts. Entwicklung der Szenarien: Aufwand ca. 10k€

10. Aufbereiten und Dokumentieren der Artefakte (wofür gebraucht, wie verarbeitet)

11. Erstellen von dokumentierten Bibliotheken (mit Javadoc, aber auch ergänzender Online-Dokumentation)

12. Erstellen einer Online-Dokumentation zur Nutzung des P23R-Client-Toolkit mit folgenden Inhalten:

    - Wozu ist das P23R-Client-Toolkit (PCT) gut?
    - Wie nutzt man das PCT?
    - Wie sieht ein typischer Bearbeitungsprozess für den Anwender des P23RClient aus?
    - Komplette Tour durch das PCT mit einer Schritt-für-Schritt-Anleitung mit dem Ziel einer lauffähigen P23R-Client-Anwendung. Dabei ist die Möglichkeit vorzusehen, die tabellarische Darstellung und die Navigation zwischen verschiedenen Formularen zu überspringen. Die Erstellung von PDF-Dateien mit einem geeigneten Werkzeug sollte ebenfalls in einer Schritt-für-Schritt- Anleitung erläutert werden.
    - Dokumentation der einzelnen Schritte: „Was ist zu tun?“ Berücksichtigen, welche Fragen sich der Entwickler dabei stellen könnte oder welchen Schwierigkeiten er sich ausgesetzt sehen könnte und anbieten von Lösungsvorschlägen und Links auf die Dokumentation der eingesetzten Werkzeuge (Ajax- Framework etc.)
    - Dokumentation aller notwendigen Artefakte (Bibliotheksfunktionen, Stylesheets etc.)
    - Downloads bzw. Download-Links auf alle notwendigen Bestandteile des PCT

(Dokumentationsaufwände ca. 45k€)

13. Bereitstellung eines Client-Test-Prozessors, der die notwendige(n) Regel(n) zum Testen des P23R-Clients enthält und verwendet werden kann, um die Funktionalität des P23R-Clients mit einem Fall (Es können nicht beliebige Regeln bereit gestellt werden, da die CommunicationConnectoren sonst evtl. zum Versenden von Spam etc. verwendet werden.) zu testen. Dies umfasst alle sieben oben aufgeführten P23R-Kommunikationsfunktionen. Dass eine Regel ausgelöst und eine Benachrichtigung freigegeben oder gelöscht wurde, wird auf einer zu diesem Zweck entwickelten Log-Webseite angezeigt. Daher ist beim Auslösen der Regel eine ID mitzuliefern, die die Identifikation des jeweiligen Nutzers auch bei der Darstellung auf der Log-Webseite möglich macht. Der/die notwendige(n) Source- und CommunicationConnector(en) zur Ausführung der Regel werden ebenfalls bereit gestellt. (Aufwand ca. 25k€)

Zielstellung des P23R-Client-Toolkits (PCT) ist ein praxistaugliches, robustes und einfach zu nutzendes Toolkit, das zur Erstellung von P23R-Clients eingesetzt werden kann. Der geschätzte Gesamtaufwand für die Entwicklung und Dokumentation des PCT beträgt ca. 170 k€.
Der als Kalkulationsbasis angenommene und in der Dokumentation der Tour dargestellte Nutzungsfall ist kein „Hello World“. Er ist von Umfang und Komplexität an eine reale Umwelt-Berichtspflicht mit mindestens einem Dutzend Berichtsobjekten, die neben dem Hauptformular mindestens zwei unterschiedliche Formulare zur Darstellung der Berichtsobjekte umfassen, angelehnt. Ggf. wird auf eine reale Umweltberichtspflicht zurückgegriffen.

G. Öffentlichkeitsarbeit/ Workshop

G.1. Webseite

Das Projekt ist unter der Webseite www.p23r4flex.de für die Öffentlichkeit zugänglich gemacht. Die Projektinhalte, Aufgaben und Ergebnisse werden transparent und offen dargelegt. Wichtige Informationen über Veranstaltungen werden zeitnah bekanntgegeben. Im August 2013 wurde einen Zwischenbericht veröffentlicht. Auch dieser Abschlussbericht wird nach Freigabe zum Download bereit gestellt.

G.2. Cebit 2013/ 2014

Das BMI präsentierte auf der CeBIT 2013 (5. bis 9. März) im Ausstellungsbereich „Effiziente Prozesse“ den P23R (Prozess-Daten-Beschleuniger). Zusammen mit den Forschungsnehmern beteiligte sich das UBA mit der Außendarstellung des Projektes P23R4FLEX auf dem BMI-Stand (im Handlungsfeld Prozessketten im eGovernment). Während des Besuch des IT-Rates und der Stattssekretärin Cornelia Rogall-Grothe am 6.3. hatte Herr Dr. Heidemeier am UBA-Stand die Möglichkeit, ein Interview zum P23R4FLEX-Projekt geben.
Auf der Cebit 2014 (10. bis 14. März) war das P23R4FLEX-Team ebenfalls wieder auf dem BMI-Stand vertreten und stellte zusammen mit dem Auftragnehmer ENDA die Ergebnisse des Projektes vor. Des weiteren gab es Auskunft zum nun folgendem Projekt - P23R im Wirkbetrieb bei der BASF.

G.3. P23R4FLEX-Workshop 2013

Die Abschlussveranstaltung des Projektes „P23R4FLEX“ fand auf Einladung des UBA am 28.11.2013 von 10:30 Uhr bis 16 Uhr in der „Vertretung des Landes Sachsen-Anhalt beim Bund“ in Berlin statt. Es waren 63 Personen aus Behörden und Unternehmen anwesend. Der Vizepräsident des Umweltbundesamtes hielt die Eröffnungsrede, der stellvertretende IT-Direktor des BMI, Herr Batt, betonte die sehr gute Zusammenarbeit mit dem UBA im Projekt.
In sieben Fachvorträgen (siehe Agenda) wurden den Behörden und Unternehmen das P23R-Prinzip verdeutlicht und anhand der technischen Ergebnisse des Projektes P23R4FLEX der praxisnahe Wirkbetrieb gezeigt. Die erstmals programmierten und frei verfügbaren P23R-Regeln mit integrierter Qualitätssicherung wurden erläutert und die Werkzeuge zur Programmierung vorgestellt. In mehreren Diskussionsrunden mit den Teilnehmern wurden die Potentiale des Einsatzes von P23R im Umweltbereich besprochen.
Alle Vorträge, die Poster sowie der Flyer sind auf der Webseite www.p23r4flex.de der Öffentlichkeit zur Verfügung gestellt.

10:00 - 10:30	Einlass und Registrierung	
10:30 - 10:45	Begrüßung und Einführung in das Thema der Veranstaltung	Vizepräsident des Umweltbundesamtes Dr. Thomas Holzmann
10:45 - 11:00	Elektronische Prozesse vernetzt - Moderne Verwaltung nutzt moderne Infrastrukturen	Ständiger Vertreter des IT-Direktors im Bundesministerium des Innern Herr Peter Batt
11:00 - 11:20	P23R - Ein stimmiges Konzept zur Vereinfachung der Kommunikation	Carsten Rosche Bundesministerium des Innern
	Im Anschluss ca. 5min Diskussion	
11:25 - 11:45	Kaffeepause	
11:45 - 12:05	Berichterstattung Kommunalabwasser - Der ideale Kandidat für den P23R	Dr. Joachim Heidemeier FG Stoffhaushalt Gewässer, UBA
12:05 - 12:20	§61 WHG und 91/271/EWG-Berichterstattung - Prozesse und Prüfungen für den P23R aufbereiten	Matthias Lüttgert ENDA GmbH & Co. KG
	Im Anschluss ca. 10min Diskussion für beide Beiträge	
12:30 - 13:30	Mittagspause	
13:30 - 13:50	P23R Regeln implementieren und veröffentlichen - P23R für Einsteiger	Jan Gottschick Fraunhofer FOKUS, eGovernment und Applikationen
13:50 - 14:15	Kommunalabwasser Prozesse im P23R - Live-Demo, Bilder und wenig Text	Matthias Lüttgert ENDA GmbH & Co. KG
14:15 - 14:35	Prüfungen als P23R-Regel implementieren - So wird‘s gemacht	Jeroen Gremmen Atos IT Solutions and Services GmbH
	Im Anschluss ca. 10min Diskussion	
14:45 - 15:05	Kaffeepause	
15:05 - 15:25	Mit P23R Regeln UMW Meldungen erzeugen - P23R-inside	Simon Dutkowski Fraunhofer FOKUS, eGovernment und Applikationen
	Im Anschluss ca. 5min Diskussion	
15:30 - 15:40	Summing-Up : Wohin geht die Reise	Dr. Joachim Heidemeier FG Stoffhaushalt Gewässer, UBA
15:40 - 16:00	Gemeinsame Diskussion und Austausch	

G.4. Weitere Projektpräsentationen

Während der Projektlaufzeit wurde das Projekt außerdem auch auf folgenden Veranstaltungen vorgestellt:

2012

05.09.	bilaterales Gespräch UBA – BMU, Bonn
20.11.	VCI-Veranstaltung zu BUBE-online, Düsseldorf
03.12.	VCI-Veranstaltung zu BUBE-online, Mannheim
04.12.	Forum Prozessketten, Metropolregion Rhein-Neckar, Sinsheim
20.12.	P23R-Anwenderkreistreffen, Mainz

2013

26.-27.02.	15. Leitungsgruppensitzung des Bund-Länder-Kooperationsprojektes VKoopUIS "ePRTR", Halle
16.05.	Forum Prozessketten, Metropolregion Rhein-Neckar, Mannheim
17.05.	P23R-Anwenderkreistreffen, Mannheim
06.06.	bilaterales Gespräch UBA (Deutschland) – UBA (Österreich), Berlin
10.06.	bilaterales Gespräch UBA - Landesamt für Landwirtschaft, Umwelt und ländliche Räume Schleswig-Holstein, Flintbek
24.10.	2. Workshop zu SIIFs (Structured Information and Implementation Frameworks) in der Berichterstattung zur EU-Kommunalabwasserrichtlinie, Brüssel
21.-22.11.	Bund-Länder-Arbeitskreis (BLAK) Abwasser

2014

14.01.	PRTR-IED-Workshop, Brüssel
07.02.	bilaterales Gespräch UBA – BMUB, Bonn
18.-19.03.	16. Leitungsgruppensitzung des Bund-Länder-Kooperationsprojektes VKoopUIS "ePRTR", Dessau
08.05.	Sitzung des Ständigen Ausschusses Umweltinformationssysteme (StA UIS), Bremen

G.5. Demonstrationsumgebung und Präsentation der Berichtspflichten

Im Rahmen des P23R4FLEX-Abschlussworkshops am 28.11.2013 wurden eine funktionsfähige P23R-Architektur mit zwei P23R-Prozessoren (für Bund und Länder) sowie zwei Fachsystemen für die Kommunalabwasserberichterstattung aufgebaut. Dabei liefen alle vier Systeme als virtuelle Maschinen unter VMware Workstation gleichzeitig auf einem Notebook.
Die gleiche Präsentation wurde auf der CeBIT 2014 am P23R-Stand gehalten.

Zur Simulation der beiden Fachsysteme von Bund und Bundesland kam jeweils die gleiche Codebasis zum Einsatz, die von ENDA entwickelte E-Kommunalabwasser-Anwendung, die vom Umweltbundesamt für eben diese Berichterstattung real eingesetzt wird. Zur Unterscheidung der Systeme in der Simulation wurde lediglich die Hintergrundfarbe der Anwendung beim Bundessystem von blau in gelb geändert. Jede der virtuellen Fachanwendungen für die Simulation hat ihre eigene Datenbank mit eigenem Datenbestand.
Die P23R-Prozessoren bestehen aus der Fraunhofer FOKUS Musterimplementierung in Version 3.1.0. Sie sind identisch aufgebaut und nutzen als Leitstelle (Die P23R-Leitstelle bietet P23R-Regeln zum Download an. Die Technik dahinter ist ein üblicher Webserver, der in einem speziellen Format, der T-BRS, abgelegte Dateiarchive ausliefert.) einen von ENDA gehosteten Server, unter der Adresse https://leitstelle.enda.eu/ekommu/p23r.tsl. Das hier abgelegte Benachrichtigungsregelpaket umfasst zwei Regeln:

- Berichterstattung Kommunalabwasser Anforderung
- Berichterstattung Kommunalabwasser

Beide kommen im Ablauf der Demonstration nacheinander zum Einsatz.
Die Demonstration wurde live vorgeführt; eine Videoaufzeichnung vom Ablauf (Desktopvideo) steht unter der URL http://www.p23r4flex.de/files/p23r_demo_ abschlussworkshop.avi zum Download bereit. Die einzelnen Schritte der Demonstration (s.u.) sind jeweils mit Zeitangaben innerhalb dieses Videomitschnitts versehen.
Der P23R-gestützte Ablauf der Berichterstattung nach EU-Kommunalabwasser-Richtlinie (91/271/EWG) wurde in folgenden Schritten demonstriert:

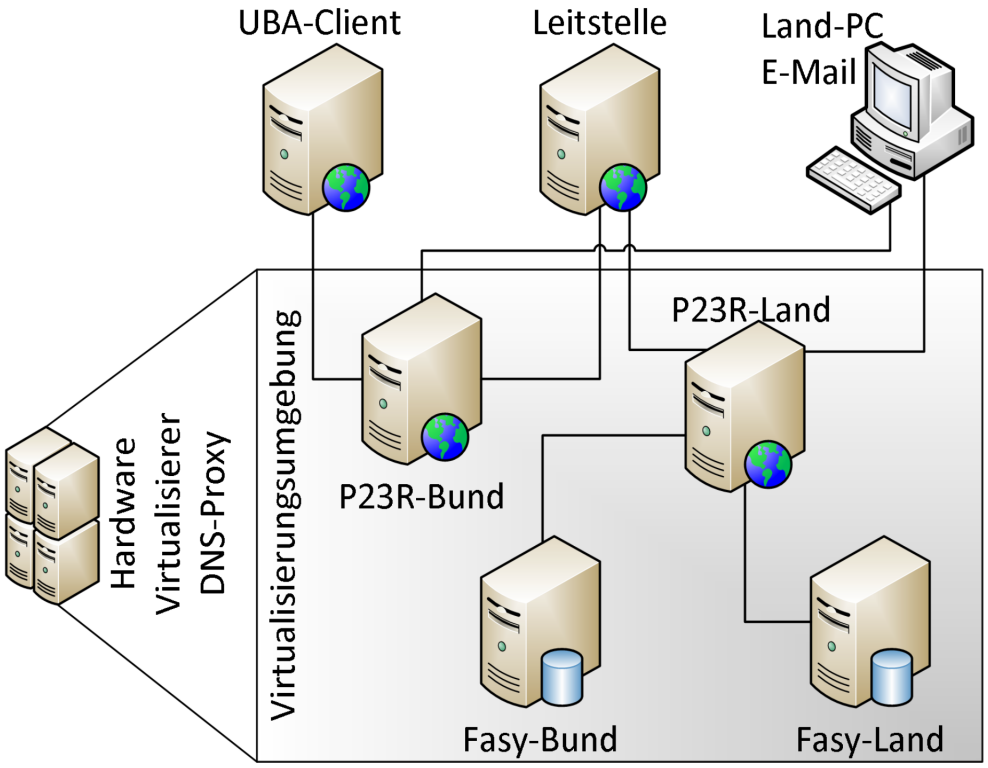


Abbildung G.1.: Simulationsumgebung der P23R-Demonstration

[0:05] Löschen der Daten in der (simulierten) Datenbank des Fachsystems Bund. Das Löschen findet automatisiert statt. Mit einer Abfrage ([0:10]) nach Agglomerationen in der Anwendung für das Berichtsjahr 2011 wird kurz demonstriert, dass die Bund-Anwendung nun „leer“ ist.

[0:37] Zeigen der vorliegenden Daten im Fachsystem Land: Die gleiche Abfrage, auf dem Landessystem ausgeführt, zeigt gut 4400 Agglomerationen.

[1:36] Anforderung der Berichterstattung: Hier kommt die erste P23R-Regel zum Einsatz. Durch manuelles Auslösen der Regel „Berichterstattung Kommunalabwasser Anforderung“ wird beim Bundes-P23R eine E-Mail an die Landesbehörden der 16 Bundesländer – in der Demo auf ein einzelnes reduziert – generiert, die zur Abgabe der Berichterstattung auffordert. Diese Regel benötigt keine Quelldaten außer den in der auslösenden Nachricht angegebenen Daten; wesentlich sind das zu berichtende Jahr und das Abgabedatum für die Berichterstattung. In der Demonstrationsumgebung kommt der generische P23R-Client der Fraunhofer FOKUS-Musterimplementierung zum Einsatz. Dieser ist für alle Regeln benutzbar, aber dadurch notwendig schlecht auf spezielle Arbeitsgebiete zugeschnitten. Insbesondere sind die Daten der auslösenden Nachricht von Hand in einen dargestellten XML-Quellcode einzutragen [2:43]; dies ist einem Fachnutzer in einer realen P23R-Umgebung nicht zuzumuten. Für solche realen Umgebungen besteht zukünftig Entwicklungsbedarf:

1. zur Entwicklung der nötigen Quellkonnektoren und 2. zur Schaffung benutzerfreundlicher Bedienoberflächen. Nachdem die Nachricht verarbeitet wurde, muss die erzeugte Benachrichtigung noch frei gegeben werden. Dies erfolgt ebenfalls im P23R-Client [2:56].

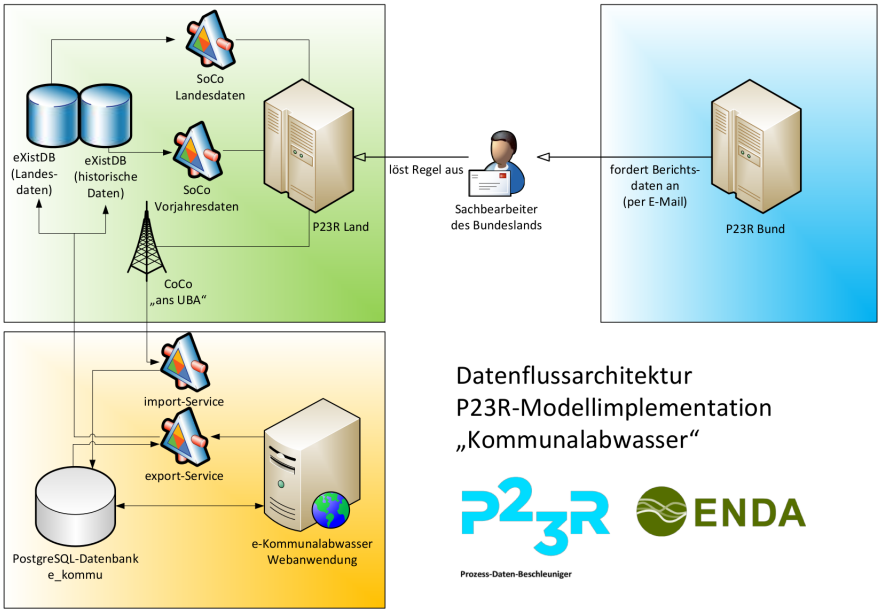


Abbildung G.2.: Datenfluss der P23R-Demonstrationsumgebung. Das Fachsystem Land ist zur Übersichtlichkeit weg gelassen; seine Daten werden in die dargestellte eXist- Datenbank „Landesdaten“ exportiert.

[3:26] Zeigen der generierten E-Mail: Der erfolgreiche Versand der Aufforderungs-E-Mail wird dann in einem Webmailsystem gezeigt.

[4:05] Länder liefern Daten (I), hier sind die Daten ungültig: Jetzt werden die Länder aktiv, stellvertretend eines, dessen simuliertes Fachsystem in Schritt 2 bereits gezeigt wurde. Dazu löst der Anwender des Landes im Landes-P23R die andere Regel, „Berichterstattung Kommunalabwasser“, aus. Diese benötigt als Parameter die Bundesland-Kennung, das Jahr der aktuellen Berichtsperiode und das Jahr der Vorperiode. Dieses wird benötigt, um (vom Bund abgelegte) Vergleichsdaten anzufordern, die in die Qualitätssicherung innerhalb der Regel eingehen. Die hier ausgelöste Regel ist deutlich komplexer als die Anforderungsregel. In ihrem Ablauf werden die aktuellen und die Vorjahresdaten abgerufen und eine Reihe von Plausibilitätsprüfungen durchlaufen. Abhängig vom Ergebnis wird im Fehlerfall nur eine E-Mail an den Anwender im Land generiert oder im Erfolgsfall sowohl die EMail generiert als auch ein Import der Daten ins Fachsystem Bund durchgeführt. Auch dieses Ergebnis muss wieder im P23R-Client frei gegeben werden, bevor E-Mails oder Importe verschickt

werden. Im ersten Durchlauf dieser Demonstration sind die Daten ungültig und bestehen daher nicht die Qualitätsprüfung in der Regel.

[6:15] Zeigen der generierten E-Mail: Die generierte E-Mail wird wiederum im Webmailsystem gezeigt. Das XML-Ergebnis der Regel wird als Anhang mit der Mail versandt und enthält eine Reihe von Warnungs- und Fehlermeldungen.

[7:14] Ein kurzer Blick ins Fachsystem Bund zeigt, dass keine Daten importiert wurden.

[7:33] Länder liefern Daten (II), hier sind die Daten gültig: Im zweiten Versuch wird die Berichterstattungsregel für ein anderes Bundesland, für das korrekte Daten hinterlegt wurden, ausgelöst und die erzeugte Benachrichtigung frei gegeben.

[8:40] Zeigen der generierten E-Mail: Durch die Freigabe der Benachrichttgung wurde wiederum eine E-Mail mit dem Prüfungsergebnis verschickt, die hier angezeigt wird.

[9:25] Zeigen der importierten Daten Durch erneutes Durchführen der Abfrage nach Agglomerationen auf dem Fachsystem Bund wird demonstriert, dass die Daten – 2 Agglomerationen – im Fachsystem des Bunds angekommen sind. Dass hier nur zwei Agglomerationen angezeigt werden statt der ursprünglichen 4405, liegt daran, dass nur die Agglomerationen des einen Bundeslands berichtet wurden.

Die Folien zu der beschriebenen Demonstration (und der anderen Vorträge des P23R4FLEX Abschlussworkshops) sind auf der Webseite des Projekts unter http://www. p23r4flex.de/abschlussworkshop abgelegt

H. Abschlussbericht des Auftragnehmers Fraunhofer Fokus / Atos

Im folgenden Kapitel wird der Abschlussbericht des Auftragnehmers Fraunhofer FOKUS (Subauftragnehmer ATOS) auszugsweise wiedergegeben (Stand: 24.03.2014).

H.1. Einleitung

Im Rahmen des aus eGovernment Mitteln des Bundes über das Bundesministerium des Innern finanzierten Projekts des Umweltbundesamtes (UBA) „Betriebliche Umweltdatenberichterstattung nach dem P23R-Prinzip - Nutzung standardisierter Berichtskomponenten“ (kurz: P23R4FLEX) waren Fraunhofer FOKUS als Auftragnehmer und Atos als Unterauftragnehmer von Juni 2013 bis Februar 2014 involviert. Das Projekt basiert auf den beiden Vorläuferprojekten „XUBetrieb“ (UBA) und „P23R“ (Prozess-Daten-Beschleuniger, BMI), deren Ergebnisse in diesem Projekt zur Abwicklung von Umweltmeldepflichten zusammengeführt, weiterentwickelt und in der Praxis getestet wurden.
Ziel des Prozess-Daten-Beschleunigers P23R (http://www.p23r.de) ist es, die Kommunikation zwischen Wirtschaft und Verwaltung sowie Verwaltungsverfahren an sich zeitlich und organisatorisch zu optimieren und damit Bürokratiekosten deutlich zu senken. P23R setzt dabei auf eine Infrastruktur, die die Datenhoheit bei den Unternehmen belässt und diese in die Lage versetzt, ihren gesetzlich vorgeschriebenen Berichtspflichten nachzukommen, wobei ggf. bereits bestehende Lösungen ergänzt und erweitert werden können. Gleichzeitig sollen Unternehmen von Detailregelungen, Plausibilitätsprüfungen und Aktualisierungen entlastet werden. Behörden erhalten durch die Nutzung effizienterer, weil medienbruchfreier Prozessketten schneller Daten in deutlich höherer Qualität.
In Vorbereitung einer breiten Einführung und Anwendung des P23R-Prinzips in der Kommunikation zwischen Betreiber und Behörde wurde im Projekt P23R4FLEX das P23R-Prinzip an Hand von zwei Umweltmeldepflichten modellhaft in der Praxis erprobt und weiter entwickelt und die Übertragbarkeit der erzielten Ergebnisse auf weitere Projekte durch die Erstellung eines Entwicklerhandbuchs und eines Konzepts zur Wiederverwendung unterstützt. Parallel zu P23R4FLEX wurde von Fraunhofer FOKUS die konzipierte P23R-Infrastruktur konkretisiert und die P23R-Musterimplementierung sowie das P23R-Entwicklerportal den sich aus P23R4FLEX er-gebenden Praxisanforderungen angepasst. Neben Fraunhofer FOKUS und ATOS hat der Partner ENDA als weiterer Auftragnehmer des UBA den P23R-Prozessor in die Testumgebung des UBA integriert und die Umsetzung der Anwendungsszenarien mit den beiden Umweltmeldepflichten umgesetzt. Dies umfasste auch die Definition der Plausibilitätsregeln zur Qualitätssicherung (QS) der Meldedaten. Auf Basis der entwickelten P23R-Regeln wurde geprüft, ob die erstellten Meldedaten die definierten QS-Anforderungen er-

füllen. Es konnte gezeigt werden, dass sich diese Prüfungen in der aktuellen Version der P23R-Regelsprache abbilden lassen. In einem Konzept zur Baustein-unterstützten Erstellung von P23R-Benachrichtigungsregeln werden beispielhaft Möglichkeiten für eine einfachere Erstellung von Regeln aufgezeigt.
Der AG beabsichtigt, alle Projektergebnisse unter einer freien Lizenz im Internet zur Verfügung zu stellen.

H.2. Ziel und Aufgabenstellung im Rahmen der P23R-Pilotvorhaben

H.2.1. Projektziele

Ziel des Projekts P23R4FLEX war es, eine wirkbetriebsfähige Lösung zur P23R-konformen Umsetzung zweier Meldeprozesse (XUKommunalabwasser, §61 WHG) aufzubauen.
Das P23R-Pilotvorhaben „Be-triebliche Umweltdatenberichterstattung nach dem P23R-Prinzip unter Nutzung standardisierter Berichtskomponenten (kurz P23R4FLEX)“ erforderte daher ein enges Zusammenwirken der Akteure im UBA mit den Akteuren aus den P23R-Vorhaben (UBA, ENDA, Fraunhofer FOKUS, Atos). Zur Zielerreichung des Projekts P23R4FLEX waren als Unterstützungsleistung von Fraunhofer FOKUS und Atos a) P23R-Regeln zur Umsetzung zweier Meldeprozesse zur P23R-konformen Umsetzung zu erstellen, zu testen und mit Hilfe der P23R4FLEX Projektleitstelle bereitzustellen. b) die P23R Musterimplementierung, heute FOKUS P23R-Pilotlösung, auf die im Rahmen des Pilotvorhabens analysierten Berichtsprozesse anzuwenden und c) dabei durchgängig XUBetrieb-konforme Nachrichten für die Kommunikation zu erzeugen und zu versenden.

H.2.2. Aufgabenstellung

Die Unterstützungsleistung von Fraunhofer FOKUS und Atos beziehen sich auf die folgenden Leistungen der Arbeitspakete im Projekt P23R4FLEX:
AP 4 P23R-konforme Umsetzung von XUKommunalabwasser

- 4.1 P23R-konforme Umsetzung (Regelerstellung), Zeitraum: Mai-August 2013 in Kooperation mit Regeltestern
- Meilenstein: P23R-Regeln für P23R-XUKommunalabwasser bis 15.08.13 beide Regeln laufen in Testumgebung zum 10.10.13

AP 6 Test P23R-XUKommunalabwasser und Entwicklung zum wirkbetriebsfähigen Prozess

- 6.2 Entwicklung zum wirkbetriebsfähigen Prozess (Regelerstellung) im Juli/August in Kooperation mit Regeltestern
- Meilenstein: wirkbetriebsfähiger P23R-XUKommunalabwasser-Prozess bis 06.09.2013 beide Regeln laufen in Testumgebung zum 10.10.13

AP 7 §61 wird P23R-konform umgesetzt (für 1 kooperierendes Bundesland)

- 7.1 P23R-konforme Umsetzung §61 WHG (Regelerstellung) im September 2013 in Kooperation mit Regeltestern,

- Meilenstein: wirkbetriebsfähiger P23R-§61WHG-Prozess bis 31.10.2013, beide Regeln laufen in Testumgebung zum 10.10.13

AP 9 Analyse der implementierten Prozesskette und Konzeption wiederverwendbarer Bausteine

- 9.1 Analyse der implementierten Prozesskette und Konzeption wiederverwendbarer Bausteine und Regelkomponenten
- Meilenstein 1: abgestimmter Entwurf der Spezifikation für die Komponentenbibliothek bis 12.11.2013,
- 9.5 Entwurf eines nutzfreundlichen Handbuches für die Regelerstellung in Kooperation mit Regeltestern
- Meilenstein 2: 9.5 Nutzfreundlichen kompaktes Handbuches für die Regelerstellung Erstellung des Handbuches bis 24.12.2013 1. Draft; Veröffentlichung der überarbeiteten Version Ende März 2014 (Version 1.0)

Annahmen:

- Die Leistungsannahmen basieren auf dem Abschlussbericht zum Projekt „P23R4FLEX – Betriebliche Umweltdatenberichterstattung nach dem P23R-Prinzip – Nutzung standardisierter Berichtskomponenten“ Finale Version vom 30.April 2013.

H.2.3. Fraunhofer FOKUS Leistungen im Rahmen der P23R-Pilotvorhaben

Im Rahmen der Unterstützung von P23R-Pilotvorhaben durch das BMI und in umfassender Eigenleistung hat Fraunhofer FOKUS die P23R-Infrastrukturen (Entwicklerportal, Projektleitstelle) und die FOKUS P23R-Pilotlösung für das P23R-Pilotvorhaben P23R4FLEX auf Basis von im Projekt identifizierten Anforderungen weiter ausgebaut und mit dem erforderlichen Support bereitstellt.
Wir weisen darauf hin, dass die von Fraunhofer FOKUS für das P23R-Pilotvorhaben bereitgestellte FOKUS P23R-Pilotlösung eine „Musterimplementierung“ ist und über den diesbezüglichen Funktionsumfang sowie entsprechende Betriebsfähigkeit verfügt - in Abgrenzung zu einer professionellen/ kommerziellen Softwarelösung mit z.B. umfassenden Wartungs- und Supportverträgen. Dies gilt ebenso für den Entwicklungsstand des P23R-Entwicklerportals und der genutzten Projektleitstelle.
Funktionsfähigkeit bedeutet für die P23R-Pilotvorhaben, dass eine fertig konfigurierte und getestete Instanz dauerhaft läuft und die getesteten P23R-Regeln im Projekt-Dauerbetrieb nachhaltig zeitgesteuert oder manuell ausgelöst und sicher ausführen werden. Betriebsfähigkeit bedeutet NICHT, dass die FOKUS P23R-Pilotlösung beispielsweise tolerant auf alle möglichen Veränderungen der Umgebung vorbereitet ist und neue P23R-Regeln mit erweitertem Funktionsumfang ohne Update der P23R-Instanz ausführen oder eine unbegrenzte Menge an Daten verarbeiten kann.

H.3. Relevante Erfahrungen im Projekt P23R4FLEX

H.3.1. Erstellung von P23R-Regeln und Nutzung Entwicklerportal & Testumgebung

- Ein wichtiges Ziel für P23R4FLEX war die Integration von QS-Prüfungen innerhalb der P23R-Regeln. Testfälle als Basis zum Testen von Benachrichtigungs-

regeln sind notwendig, aber in ihrer Granularität noch zu grob, d.h. der Aufwand, für alle Feinheiten einen Testfall zu schreiben, ist unverhältnismäßig hoch. Daher wurde die Möglichkeit geschaffen, in eine Benachrichtigungsregel semantische Validierungsfunktionen zu integrieren, die ihre Testergebnisse im Protokoll beim Generieren einer Benachrichtigung detailliert dokumentieren. Die Ergebnisse der einzelnen Tests werden im Testprotokoll beim Abarbeiten eines Testpaketes aufgeführt, so dass man in dieser Kombination auch sehr fein granular eine Benachrichtigungsregel testen kann, um potentielle Fehler einfacher lokalisieren zu können. Somit stehen aus der Erfahrung mit P23R4FLEX dem Regelersteller im P23R-Entwicklerportal (Version 0.8 / März 2014) eine praxisnahe Testumgebung für Unittests bzw. „Behaviour Driven Development“ zu Verfügung.

- Bereits erstelle Artefakte können teilweise wieder verwendet werden. Nach der initialen Regelerstellung konnte aufgrund der gewonnen Erfahrungen bei allen Projektbeteiligten die Erstellung der 2. Regel in der gleichen Domäne in relevant viel kürzerer Zeit erfolgen. Dies betrifft einerseits die fachliche Abstimmung und andererseits die Regelerstellung selbst durch Wiederverwendung von Datenmodellen, Validierungsregeln und Konnektoren.

- Das Projekt hat gezeigt, dass das Entwicklerportal das Paketieren, Hochladen und teilweise Testen von Regeln und Benachrichtigungsregelgruppen unterstützt. In Bezug auf die Regelerstellung bietet das Portal Templates für die erforderlichen Artefakte zur Definition der Regelgruppen und Pivot-Datenmodelle sowie die Möglichkeit die Definitionen anzusehen und im Webportal zu editieren.

- Der Einsatz einer eigenen offline XML-Entwicklungsumgebung durch die Entwickler für die Entwicklung von Regeln ist notwendig.

- Entwickler benötigen bei der Erstellung von Benachrichtigungsregelgruppen die Möglichkeit, diese ohne einen vor Ort installierten P23R-Prozessor zu testen. Entsprechend gibt es nun die Möglichkeit, Benachrichtigungsregelgruppen in einer Projekt-Leitstelle zu veröffentlichen, die Testpakete mittels des Test-P23R-Prozessors direkt im P23R-Entwicklerportal auszuführen und sich die Ergebnisse anzeigen zu lassen.

- Der Test-P23R-Prozessor lieferte während der Projektlaufzeit noch nicht ausreichende debugging-Informationen, diese sollten zur Verbesserung der Test-Möglichkeiten zukünftig ausgebaut werden. So sollte bspw. nach einem Test sowohl das Ergebnis als auch die Lokalisation der Fehlerquelle als „menschenlesbarer“ Dialog dem Bearbeiter zugänglich gemacht werden

- In der Zukunft muss das Entwicklerportal in die Lage versetzt werden, Regeln zuverlässig und umfassend auf syntaktische Korrektheit zu prüfen, denn erst syntaktisch korrekte Regeln können auf ihre semantische Korrektheit geprüft werden. So kann der P23R-Prozessor vor nicht validen Benachrichtigungsregeln geschützt werden.

- Um Verantwortlichkeiten klar zu trennen, werden die Projekte bisher je nach Zweck (Benach-richtigungsregelgruppe, Pivot-Modell etc.) komplett atomar aufgeteilt. Bei der Regelerstellung werden immer ein Datenmodell, eine Benachrichtigungsregelgruppe, ein Testsatz, und eine Projektleitstelle benötigt. Die Erfahrung im Projekt P23R4FLEX hat gezeigt, dass eine Kombination dieser einzelnen Projekte in einem Kombi-Projekt sinnvoll ist, so dass dann auch nur noch ein einzelnes GIT-Repository benötigt wird.
- Die Nutzung externen Speichers in Form von GIT hat sich als praktikabel erwiesen. Allerdings muss für die Regelersteller zukünftig die Handhabung durch verbessertes Fehlerhandling und intelligente Funktionen, bspw. beim initialen Einrichten des GIT, vereinfacht werden.

H.3.2. Entwicklerhandbuch / Checkliste

- Das Handbuch Version 1.0 (März 2014) für Einsteiger zum Erstellen von Benachrichtigungsregeln setzt Vorwissen zu P23R im Allgemeinen – und der T-BRS im Speziellen – voraus, da es einen Einblick in die praktische Benachrichtigungsregelerstellung geben soll. Eine Einarbeitung in die grundlegenden P23R-Architekturdokumente ist für Entwickler daher sinnvoll. Das Handbuch bezieht sich auf die Version 0.8 (März 2014) des Entwicklerportals.
- Es gibt unterschiedliche, subjektive Vorstellungen zur Reihenfolge beim Erstellen von Regeln. Im Handbuch wird nur die Variante A beschrieben, die die Entwicklung der Benachrichtigungsregelgruppe mit der Modellierung der Zieldaten beginnt. Das hat den Vorteil, dass vor Implementierung der Transformationsschritte die benötigten Daten genau bekannt sind. Variante B zieht die Entwicklung einer Regel vom Ausgangspunkt der Quelldatenmodelle auf.
- Die im Handbuch vorgeschlagene Anforderungsanalyse erscheint zunächst unattraktiv, ist aber sinnvoll und erforderlich, um während der Entwicklung auf den in den ersten Schritten definierten Artefakte aufbauen zu können. Dies gilt insbesondere in Hinblick auf die Verwaltung der vielfältigen Namensräume (Quelldaten, Transformationen etc.).
- Der Einstieg in das Entwickeln von P23R-Benachrichtigungsregeln stellt für neue Entwickler – wie bei allen neuen Technologien – immer eine Hürde dar. Entsprechend hat sich auch bei P23R4FLEX gezeigt, dass ein leicht lesbares Handbuch für P23R-Einsteiger wichtig, aber nicht ausreichend ist.
- Einsteiger in die Regelerstellung benötigen einen roten Faden, was konkret in welcher Reihenfolge bei der Regelerstellung gemacht werden muss, bzw. welche Schritte als nächstes abzuarbeiten sind. Entsprechend wurde in P23R4FLEX neben dem Handbuch auch eine Checkliste erstellt, die einen Überblick über alle erforderlichen Schritte liefert. Diese sollte zukünftig auf Basis weiterer Erfahrungen angepasst und verfeinert werden.
- Ein automatisierter Assistent, der den Regelersteller durch die Bearbeitungsschritte führt, wäre hilfreich, ist aber technisch sehr aufwendig.

- Das Entwicklerbootcamp hat gezeigt, dass mit der Erstellung der Beispielregel an Hand des Handbuches ein grundlegendes Verständnis für die Regelerstellung geschaffen wird.
- Im Handbuch sind die folgenden Funktionen und Handlungsempfehlungen beschreiben. Anhand der mit * gekennzeichneten Funktionen wurde im Bootcamp der Entwicklungsprozess exemplarisch durchgeführt. Dabei festgestellte Verbesserungsbedarfe wurden im Entwicklerhandbuch eingepflegt bzw. in der Version 0.8 (März 2014) des Entwicklerportals verbessert.

Wissen

- Info-Bereich
  - Anleitungen
  - Spezifikationen
  - Einsteigerhandbuch*
  - Beispiele
  - FAQ
- Testlabor
  - P23R Model

Werkzeuge

- Syntaxgesteuerte Editoren*
  - XML
  - P23R Selection
  - P23R Model
  - XQuery
  - Markdown
  - Manifest-Editor
- Projektvorlagen*
- Testumgebung*

Prozesse

- Projekt-Leitstelle*
  - Veröffentlichung der Entwicklungsergebnisse

H.3.3. Konzepte zur Wiederverwendung

- Sehr wichtig ist in allen P23R-Projekten das Sammeln von wichtigen bzw. immer wieder auf-tauchende Fragen (FAQs) bereits in der Projektlaufzeit, um so erzielte Erkenntnisse für die Nachwelt zu dokumentieren. Dies bedeutet einen Personalaufwand insbesondere für die Entwickler, der im Projektplan zu berücksichtigen ist.

- Darüber hinaus sollte wiederverwendbarer Code dokumentiert werden. Relevante Code-Beispiele sollten im Rahmen der P23R-Entwicklungsumgebung zusätzlich auch als „Codesnippets“ in die Editoren integriert werden. In P23R4FLEX hat sich gezeigt, dass die Identifikation von wiederverwendbaren „code snippets“ am besten unter Mitwirkung aller Beteiligten schon während der Entwicklung erfolgt. So sollte auch in Zukunft, bspw. in Rahmen von Reviews oder Entwicklerbootcamps, weiterer potentiell wiederverwendbarer Code aktiv identifiziert und geeignet als „code snippets“ oder/und FAQ dokumentiert werden. Ferner muss in Folgeprojekten auch entsprechende Personalressourcen für die Entwickler zur Erstellung von „code snippets“ einkalkuliert werden, da nur diese in der Lage sind, die betroffenen Code-Teile in einer generischen Form (Code-Ausschnitt) bereit zu stellen.

- Die Validierung von Daten kann als eine zentrale Fähigkeit von P23R eingesetzt werden. Da-her ergibt sich die Frage, wie man syntaktische Validierungen (Schemaprüfung), aber vor allem auch semantische Validierungen einfacher für eine Benachrichtigungsregel erstellen kann. Einen vielversprechenden Ansatz hat P23R4FLEX mit den OCL-basierten Validierungen aufgezeigt. Eine P23R-spezifische Teilmenge von OCL in Rahmen von P23R Script könnte die Erstellung und Pflege von semantischen Validierungen deutlich vereinfachen. Ein UML-Komponenten-Diagramm ist zur Visualisierung der für die Benachrichtigungsregel benötigten P23R-Infrastruktur zweckmäßig, um die benötigten Konnektoren zu identifizieren und deren Einsatz zu dokumentieren. Die Weiterentwicklung einer einheitlichen Dokumentation für P23R-Benachrichtigungsregelgruppen und Pivot-Teildatenmodelle ist sinnvoll, um einerseits eine einheitliche Qualität bei der Dokumentation zu gewährleisten, aber auch um andererseits Erfahrungen bei der Analyse und Beschreibung derselben weiter zugeben.

H.3.4. PM / QM Lessons learned:

- Erfolgskritisch für ein Projekt ist die interne Kommunikation. Eine Maßnahme zur Verbesserung der bilateralen und übergreifenden Kommunikation zwischen den jeweiligen Projektbeteiligten, die an gemeinsam zu bewältigenden Aufgaben arbeiten, kann die frühzeitige gemeinsame Durchführung von Workshops zu diversen organisatorischen und fachlichen Herausforderungen im Projekt sein. Entsprechende Maßnahmen, die auch der Verbesserung von Verständnis, Stimmung und Motivation dienen, sollten im Projektbudget eingeplant werden. Ein gutes Beispiel ist das Entwicklerbootcamp, das im Februar 2014 mit dem Auftraggeber, ENDA und FOKUS durchgeführt wurde. Grundsätzlich ist zu beachten, dass Workshops zeit- und personal- und mithin kostenintensiv sind; sie sind daher sehr gezielt und fokussiert einzusetzen.

- Bei den Projektbeteiligten existierten unterschiedliche Annahmen in Bezug auf den existierenden Funktionsumfang der vorhandenen P23R-Infrastruktur. Daraus resultierten ein unterschiedliches Verständnis und unterschiedliche Erwartungshaltung in Bezug auf Leistungsumfang und -tiefe bei einigen Arbeitspaketen. Im Rahmen der Vorgespräche und Verhandlung wurden Aufwandsreduzierungen insbesondere in den Bereichen PM/QM (18 >5), „Wiederverwendung von Bausteinen“ (18>12) und Handbuch (20>10) vorgenommen. Dennoch erforderte sowohl die Erstellung der P23R4Flex-Ergebnisse als auch PM/QM später Mehraufwände auf allen Seiten. Es bestanden unterschiedlichen Einschätzungen in Bezug auf Leistungsbestandteile bzw. Ergebnistiefe bei den Projektpartnern.
- Ressourceneinsatz und Timing
  - Höhere Aufwände bei der Spezifikation der Anforderungen und Pivot-Modell Schemata sowie für die Erstellung der Artefakte (deutlich mehr als geplant) für das P23R-Regelpaket XUKommunalabwasser, da für die Erstellung der ersten Regel umfassen-des Wissen und Erfahrungen aufzubauen sind.
  - Höherer Aufwand bei der Unterstützung der Fa. ENDA zur Einarbeitung und Nutzung des Entwicklerportals.
  - Höherer Aufwand bei der Anpassung der P23R-Pilotlösung durch die Anforderungen an die Testumgebung / Test-P23R sowie die Änderungen an den Skripten (s.u.) und Unterstützung von ENDA bei der Inbetriebnahme der P23R-Pilotlösung (z.B. Konnektoren)
  - Höherer Aufwand bei der Spezifikation der Anforderungen und Pivot-Modell Schemata für XUKommunalabwasser auf Grundlage des Abschlussberichts zum Projekt „P23R4FLEX – Betriebliche Umweltdatenberichterstattung nach dem P23R-Prinzip – Nutzung standardisierter Berichtskomponenten“
  - Deutlich geringer Aufwand (als ursprünglich geplant) bei der Erstellung der Artefakte für das P23R-Regelpaket zur Umsetzung von § 61 WHG
  - Mehraufwand in Bezug auf die Erstellung des Entwicklerhandbuchs insbesondere durch die parallele Entwicklung und Adaption der Funktionen der in Entwicklung befindlichen Entwicklerumgebung mit den Beschreibungen im Handbuch, den Screenshots und den Ausführungen zu den FAQs vor dem Hintergrund des gekürzten Budgets.
  - Durch diese Mehraufwände sind auch zeitliche Verzögerungen in Bezug auf den 1. Projektplan entstanden, die jedoch eine Präsentation der Projektergebnisse im November nicht behinderten.

H.3.5. Durchgeführte zusätzliche Leistungen

Durch FhG FOKUS wurden folgende Leistungen zusätzlich erbracht:

- Parallel zu den Entwicklungen im P23R4Flex-Projekt musste seitens Fraunhofer FOKUS auch auf Basis der Anforderungen aus dem P23R4FLEX die P23R-

Infrastruktur ausgebaut werden, um das Erstellen und Testen von Regeln zu ermöglichen. Dies umfasst insbesondere folgende Themenfelder:

- Ausbau des P23R-Entwicklerportals
  - Anpassung des Entwicklerportals an die erweiterte Funktionalität der Testumgebung (Ausgabe der Testergebnisse)
  - Ausbau des Entwicklerportals an erforderliche Funktionen und Handhabung für die Regelentwickler
- Erstellung und Ausbau der P23R-Testumgebung
  - Die nach der Testspezifikation vorgesehenen Testmöglichkeiten waren nicht ausreichend. Deshalb wurden die Erstellung der Testprotokolle entsprechend verbessert. Zu den Verbesserungen gehörte vor allem der Detaillierungsgrad der Testprotokoll-Referenzen zu Skript, Zeile, Spalte sowie zu Fehlertypen, etc.
- Ausbau der FOKUS P23R-Pilotlösung in Bezug auf besondere Anforderungen aus dem Projekt P23R4FLEX
  - Die strikte Ordnung erst Selektion, dann Transformation, wurde geöffnet. Bei den einzelnen Verarbeitungsschritten können sich nun Transformationen und Selektionen abwechseln.
  - Des Weiteren wird der Zuständigkeitsfinder als regulärer „öffentlicher“ Quelldatenkonnektor betrachtet und wird nicht mehr über eine gesonderte Web Service Schnittstelle angesprochen.
  - Eine weitere Erweiterung ist das Referenzieren von Artefakten innerhalb einer Regel bzw. Modelpaketes. Dadurch können Skripte oder Daten, die in einer Regel mitgeliefert werden (z.B. Tabellen mit Adressen), einfach über imports oder includes modularisiert werden.
  - Dem Wunsch nach Gliederung und Separation der Daten von den Skripten wurde nachgekommen. Dies erforderte eine Änderung bei der Verarbeitung von Skripten.
  - Verbesserung der P23R Admin Konsole zur einfachen Konfiguration der P23R Pilotinstanz.
  - Verbesserung der Modularisierung von Skripten und Daten innerhalb von Modell- und Regelpaketen.
  - Auf Basis der Erfahrungen wurden auch in der P23R-Architektur und in den P23R-Spezifikatione Änderungen erforderlich, die bis Ende März 2014 in der P23R-Dokumentation eingepflegt werden.