Nutzung von Webservices am Beispiel eines Pseudonymisierungsdienstes

Steffen Richter
DZNE Rostock/Greifswald

Ellernholzstr. 1-2
17487 Greifswald
Steffen.richter@dzne.de

René Walk
Universitätsmedizin Greifswald
Institut für Community Medicine
Walther-Rathenau-Str. 48
17475 Greifswald
Rene.walk@uni-greifswald.de

Zusammenfassung

Im Rahmen medizinischer Forschungsprojekte spielt die korrekte Pseudonymisierung der Probanden eine wichtige Rolle. An der Universität Greifswald wurde dazu eine professionelle Infrastruktur aufgebaut, die ihre Funktionen über Webservices bereitstellt. Der Beitrag zeigt nach einer allgemeinen Einführung in das Thema Webservices, wie ein solcher Dienst verwaltet wird und im Rahmen von SAS-Programmen zur Datenaufbereitung eingebunden und genutzt werden kann.

Schlüsselwörter: Webservice, Pseudonymisierung, SOAP, XML

1 Einführung in Webservices

1.1 SOA – Service-orientierte Architekturen

Etwa um die Jahrtausendwende sahen sich viele Unternehmen und Institutionen mit dem Problem konfrontiert, dass sich ihre über die Jahre gewachsenen IT-Systeme als zu starr und unflexibel erwiesen. Oft basierten diese auf veralteten Technologien und Geschäftsanforderungen. Häufig wechselnde Bedingungen im globalen Markt erfordern aber Flexibilität und Modularität der Geschäftsprozesse und damit auch der IT-Systeme. Es war also ein Ansatz gefragt, der es der IT erlaubt, sich ändernde Geschäftsprozesse schnell und flexibel zu unterstützen. Daraus entstand die Idee, große, monolithische Anwendungen im Sinne eines Baukastensystems in kleine, spezialisierte Einheiten aufzubrechen. Diese selbständig betriebenen, verwalteten und gepflegten Softwareeinheiten, die ihre Funktion über eine implementierungsunabhängige Schnittstelle kapseln, werden als Services bezeichnet. Und dementsprechend versteht man unter einer *Service-orientierten Architektur* (SOA) eine unternehmensweite IT-Architektur mit Services (Diensten) als zentralem Konzept [1].
Als ein Architekturprinzip auf Unternehmensebene ist SOA zunächst technologieunabhängig. Im Laufe der Entwicklung haben sich aber mehrere Technologien etabliert, um die Verbindung zwischen den einzelnen Services aufzubauen. Am weitesten verbreitet sind sicherlich die Webservices in den beiden Ausprägungsformen SOAP und REST. Es kommen aber auch Technologien wie CORBA (Common Object Request Broker

Architecture) oder MOM (Message Orientated Middleware) zum Einsatz. In sehr großen Unternehmenslandschaften wird eine SOA häufig auch auf der Basis eines ESB (Enterprise Service Bus) aufgebaut, einer Middleware, die die Verwaltung und Integration aller angebotenen Dienste übernimmt.

1.2 Webservices – SOAP und REST

Wie schon angemerkt, sind Webservices die am häufigsten genutzte Technologie zur Verbindung von Systemen in einer Service-orientierten Architektur. Der Grund dafür ist die einfache Umsetzbarkeit. Der Namensteil „Web-“ sagt es schon – Basis von Webservices sind die etablierten Technologien des World Wide Web. Webservices nutzen also auch die Transportprotokolle http oder https, die von allen gängigen Programmiersprachen und -plattformen durch Standardfunktionen gut unterstützt werden. Die Ports 80 bzw. 443 sind in der Regel an den Firewalls freigeschaltet, so dass Webservices auch über die Grenzen von Unternehmen oder Organisationen angeboten bzw. aufgerufen werden können. Mit dem Konzept des URI ist ein universelles Adressierungsschema gegeben, das es auf einfache Art und Weise erlaubt zu beschreiben, wie der Service erreicht werden kann.

Die Abbildung 1 illustriert den grundlegenden Aufbau. Eine Anwendung, die einen Service nutzen will – der Servicenehmer – adressiert den Servicegeber über den URI und schickt via http oder https seine Anfrage (Request). Der Servicegeber verarbeitet diese Anfrage und schickt ebenfalls via http / https sein Ergebnis (Response) zurück.

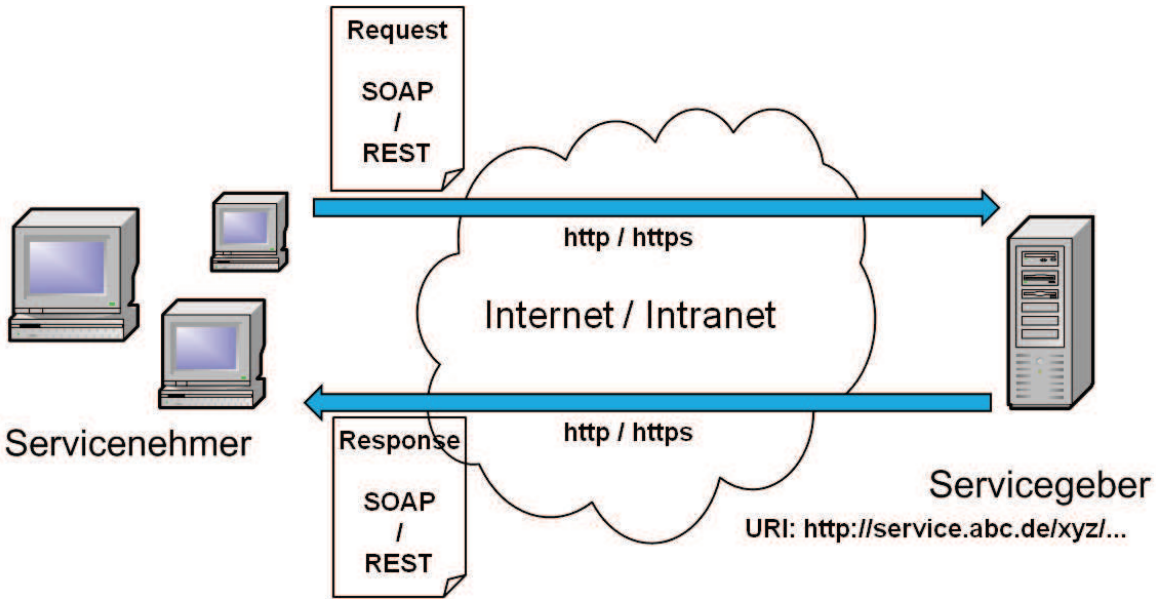

Abbildung 1: Grundprinzip von Webservices

Der Unterschied zu einer Webanwendung ist nun, dass über das Transportprotokoll http / https nicht HTML-Seiten übertragen werden, die vom Browser in einer für den Menschen lesbaren Darstellung angezeigt werden, sondern Daten in einem maschinenlesbaren Format. Dabei haben sich die beiden Formate SOAP und REST etabliert, die in den folgenden Abschnitten besprochen werden sollen.

1.3 SOAP

Hinter dem Begriff SOAP verbirgt sich ein vom World Wide Web Consortium (W3C) definierter Standard zum Austausch von Daten zwischen Systemen auf Basis von XML. Der Begriff stand ursprünglich als Abkürzung für *Simple Object Access Protokoll*, wird aber vom W3C heute so nicht mehr verwendet, da SOAP keineswegs *Simple* ist und auch nicht unbedingt nur dem Zugriff auf Objekte dient.
Das erste Dokument, mit dem man als Anwender eines SOAP-Services zu tun hat, ist die Servicebeschreibung in der sogenannten *Web Services Description Language* (WSDL). Dies ist ein XML-Dokument, das in standardisierter Form Angaben zur URI des Services sowie zu den angebotenen Operationen und deren Parameter enthält.

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions name="PSNManagerService"
        targetNamespace="http://psn.ttp.ganimed.icmvc.emau.org/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:tns="http://psn.ttp.ganimed.icmvc.emau.org/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <wsdl:service name="PSNManagerService">
    <wsdl:port binding="tns:PSNManagerServiceSoapBinding" name="gpasServicePort">
①     <soap:address location="http://localhost:4204/psn-ejb/PSNManager"/>
    </wsdl:port>
  </wsdl:service>
②<wsdl:operation name="getPseudonymForList">
    <soap:operation soapAction="" style="document"/>
③   <wsdl:input name="getPseudonymForList">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getPseudonymForListResponse">
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:types>
    <xs:schema>
④     <xs:element name="getPseudonymForList" type="tns:getPseudonymForList"/>
```

Abbildung 2: Beispielhafter Ausschnitt eines WSDL-Dokuments

Wie das Beispiel in Abbildung 2 zeigt, ist ein solches WSDL-Dokument nicht unbedingt intuitiv verständlich. Man erkennt aber bei ①, über welche Adresse der Service `PSNManager` erreichbar ist, bei ②, dass eine Operation namens `getPseudonymForList` angeboten wird, die den unter ③ genannten Inputparameter `getPseudonymForList` erwartet. Wie dieser aufzubereiten ist, wird dann bei ④ in einer Schemadefinition beschrieben, die hier nur angerissen ist. Zur einfacheren Handhabung gibt es Tools, die alle diese Informationen aus der WSDL-Beschreibung des Services auslesen und anschaulich in einer grafischen Oberfläche darstellen können, wie später gezeigt werden wird.
Verstehen sollte man als SAS-Anwender aber den Aufbau der eigentlichen SOAP-Nachrichten, denn diese müssen mit den XML-Funktionen von SAS aufbereitet bzw. verarbeitet werden. Sowohl der Request als auch der Response bestehen aus dem umfassenden Element *Envelope* ①, in das die Elemente *Header* ② und *Body* ③ eingebettet

sind. Das Header-Element ist dabei optional. Hier können Metaangaben enthalten sein, die spezielle Verarbeitungsfunktionen auf Seiten des Servicegebers oder des Servicenehmers steuern. Dazu gibt es vom W3C eine Vielzahl von Spezifikationen wie WS-Adressing, WS-Security, WS-Transaction etc. Der von uns genutzte Beispielservice nutzt diese Zusatzfunktionen nicht, so dass hier der Header leer bleiben kann. Im Body-Element erkennt man den Inputparameter `getPseudonymForList`, wie er im WSDL-Dokument beschrieben wurde.

```
① <soapenv:Envelope
            xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
            xmlns:psn="http://psn.ttp.ganimed.icmvc.emau.org/">
②    <soapenv:Header/>
③    <soapenv:Body>
        <psn:getPseudonymForList>
           <!--1 or more repetitions:-->
           <values>?</values>
           <domain>?</domain>
        </psn:getPseudonymForList>
     </soapenv:Body>
  </soapenv:Envelope>
```

Abbildung 3: Beispiel eines SOAP-Requests

1.4 REST

Der Begriff REST steht als Abkürzung für *Representional State Transfer* und beschreibt einen Architekturstil für die Kommunikation zwischen Systemen unter Nutzung der vorhandenen Infrastruktur des World Wide Web. So werden z.B. die Funktionen des Servicegebers direkt mit den HTTP-Mechanismen GET, PUT, DELETE und POST adressiert, während SOAP den Request immer mittels HTTP-POST überträgt und die eigentliche Funktion über den Methodennamen oder einen Parameter spezifiziert. REST ist auch nicht an XML gebunden wie SOAP. Häufig wird stattdessen die *JavaScript Object Notation* (JSON) zur Darstellung komplexer Datentypen genutzt, die wesentlich kompakter ist. Das zeigt ein fiktives Beispiel. Um das Pseudonym für einen einzelnen Namen zu erhalten, kann man alle benötigten Parameter in den URI des GET-Aufrufs ① packen und erhält dann als Ergebnis eine Zeile mit einem JSON-Array ②. Da der im Folgenden beschriebene Pseudonymisierungsservice aber keine REST-Aufrufe erlaubt, soll an dieser Stelle nicht tiefer auf diese Technologie eingegangen werden.

```
Request:
    GET
①   http://localhost:4204/PSNService/getPseudonymFor?
       value=Michael_Jackson&domain=BaseBall-01
Response:
②   Content-Type: text/json
    { "return": "pl_993951" }
```

Abbildung 4: Beispiel einer REST-Kommunikation

2 Der Anwendungsfall

2.1 Die unabhängige Treuhandstelle der Universitätsmedizin Greifswald

Auch im Kontext von Forschung und Entwicklung erreichen die IT-Systeme eine hohe Komplexität, so dass es sich anbietet, diese nach Prinzipien einer SOA zu organisieren und aus kleinen, wiederverwendbaren Bausteinen zusammenzusetzen. In großen epidemiologischen Studien z.B. besteht immer die Aufgabenstellung der Probandenverwaltung unter Beachtung der Anforderungen des Datenschutzes, die eine frühzeitige Trennung von personenidentifizierenden und medizinischen Daten sowie eine konsequente Pseudonymisierung oder Anonymisierung fordern. An der Universitätsmedizin Greifswald wurde dazu eine technisch und organisatorisch unabhängige Abteilung aufgebaut, die die Aufgaben eines „Datentreuhänders“ wahrnimmt, wie sie z.B. in den generischen Datenschutzkonzepten der TMF beschrieben sind [2]. Diese Aufgaben umfassen

- die Zuordnung von personenidentifizierenden Daten und entsprechenden Kennungen für Quell- und Sekundärsysteme
- die Verwaltung von Einwilligungen, Ermächtigungen und Widerrufen
- die Pseudonymisierung bzw. De-Pseudonymisierung von Daten

Beteiligt ist die Treuhandstelle z.B. an solchen Studien wie der Nationalen Kohorte [3], GANI_MED [4] oder dem zentralen klinischen Krebsregister Mecklenburg-Vorpommern [5].
Für jede der genannten Aufgaben wurde eine Software entwickelt, die unabhängig installiert und betrieben werden kann und die ihre Funktion über SOAP-Services anbietet. Diese Werkzeuge stehen unter mosaic-greifswald.de als Open-Source zur Nachnutzung zur Verfügung [6][7].

2.2 gPAS – a generic Pseudonym Administration Service

Der gPAS erlaubt das Erzeugen und Verwalten von domänenspezifischen Pseudonymen in einem beliebigen Anwendungskontext. Ausgangspunkt der Pseudonymisierung kann jede Zeichenkette sein, die den Datensatz identifiziert – Fallnummer, Name oder ein bereits vorhandenes Erstpseudonym. Die Pseudonyme werden nicht algorithmisch aus dem Identifier abgeleitet, sondern zufällig erzeugt und dann in einer Datenbanktabelle abgelegt. Somit ist sichergestellt, dass wiederholte Aufrufe des Pseudonymisierungsdienstes für einen eindeutigen Identifier immer dasselbe Pseudonym liefern und auch eine De-Pseudonymisierung möglich ist. Unter einer Domäne versteht man dabei einen konkreten Pseudonymisierungsvorgang, z.B. die Erstellung von Zweitpseudonymen für den Export der Daten an einen externen Forschungspartner. Für jede Domäne können die Parameter der Pseudonymisierung (zu verwendendes Alphabet, Länge, Präfix, Prüfzeichenalgorithmus) individuell konfiguriert werden. Die Tabelle 1 zeigt einige Beispiele dazu.

Tabelle 1: Beispiele möglicher Pseudonymisierungsparameter des gPAS

Domain	Alphabet	CheckDigitGenerator	Properties
FP_Master	Symbol31	ReedSalomonLagrange	length=8; check_digits=2
FP_Second_1	a,*,+,-,#,ß,@	ReedSalomonLagrange	length=5; check_digits=1
FP_Second_2	Numbers	Verhoeff	length=10

Dabei sei die Domäne FP_Master die Verwaltung der Erstpseudonyme in einem Forschungsprojekt FP. Mit den beiden anderen Domänen werden jeweils verschiedene Zweitpseudonyme für die Weitergabe der Daten erzeugt. In Tabelle 2 ist jetzt ersichtlich, wie der gPAS bei dieser Konfiguration den Namen Müller als ursprünglichen Identifier pseudonymisiert.

Tabelle 2: Beispiele erzeugter Pseudonyme

Domain	OriginalValue	Pseudonym
FP_Master	Müller	C80A40NX0K
FP_Second_1	C80A40NX0K	-@ß**#
FP_Second_2	C80A40NX0K	96069694219

Alle Funktionen des gPAS können sowohl über eine Web-GUI wie auch SOAP-Services genutzt werden. Die Servicefunktionen sind zu zwei Services gruppiert, den DomainManager und den PSNManager. Mit dem DomainManager können die Pseudonymisierungsdomänen eingerichtet, abgefragt und auch wieder gelöscht werden. Über den PSNManager erfolgt die eigentliche Pseudonymisierung. Er stellt Funktionen zur Pseudonymisierung und De-Pseudonymisierung von einzelnen Werten oder ganzen Listen, zur Validierung der Prüfziffer und zur Übernahme bereits bestehender Wert-Pseudonym-Paare zur Verfügung. Die zugehörige Servicebeschreibung in Form des WSDL-Dokumentes wurde ausschnittweise in Abbildung 2 gezeigt. Das folgende Beispiel wird die dort hervorgehobene Funktion `getPseudonymForList` nutzen.

3 Vorgehen – Schritt für Schritt

3.1 Beispielszenario

Nun soll konkret gezeigt werden, wie der im vorigen Abschnitt beschriebene Pseudonymisierungsdienst mit den Mitteln von SAS genutzt werden kann. Als Datenbasis für dieses Beispiel dient die Tabelle SASHELP.BASEBALL. Sie enthält Angaben zu 322 Baseballspielern, die durch ihren Namen in der Variablen NAME identifiziert werden. Für jeden Spieler sind in den weiteren 23 Variablen viele statistische Angaben zu den Leistungen in der aktuellen Saison hinterlegt, wie Anzahl der Hits, der Runs, der Home Runs usw. Es sei angenommen, dass diese Daten im Rahmen einer Studie statistisch analysiert werden sollen, wozu die Angaben zu den Spielern zu pseudonymisieren sind.

Die erzeugten Pseudonyme sollen dabei dem Schema `pl_nnnnnz` entsprechen, wobei gilt:

pl_ = konstantes Präfix für „player_“
nnnnn = zufällige fünfstellige Ziffernfolge
z = Prüfziffer nach dem Verhoeff-Algorithmus

3.2 Einrichten der Pseudonymisierungsdomäne

Um den gPAS passend zu konfigurieren, damit er Pseudonyme nach dem gewünschten Schema erzeugt, wird die Web-GUI genutzt. Die einzurichtende Domäne soll dabei BaseBall-01 heißen, was den Vorgang ausreichend beschreibt.

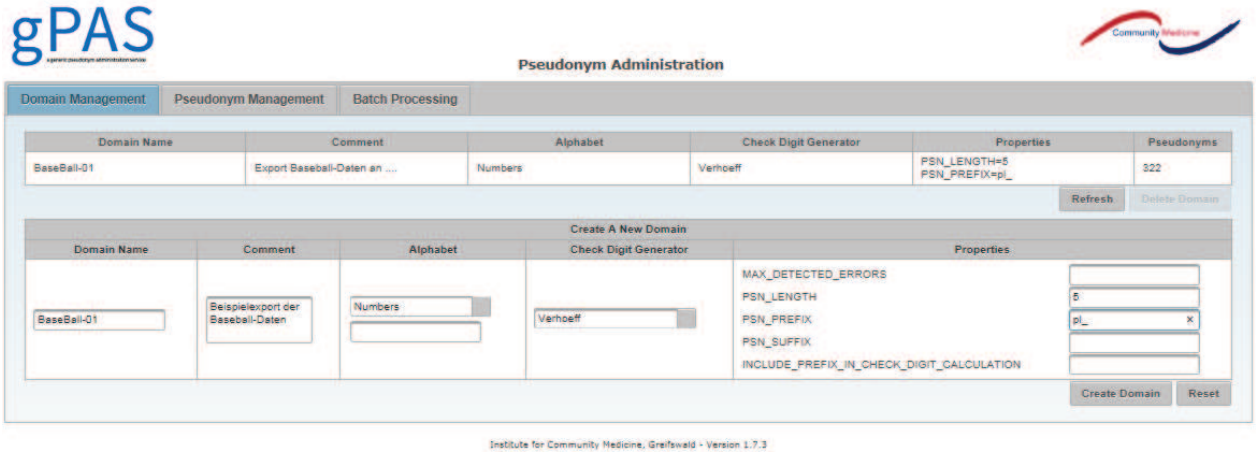

Abbildung 5: Einrichten der Pseudonymisierungsdomäne mittels Web-GUI

Man hätte das Anlegen der Pseudonymisierungsdomäne natürlich auch per Serviceaufruf vornehmen können, aber dazu müsste man den folgenden Kapiteln vorgreifen, in denen solch ein Aufruf aus SAS heraus beschrieben wird.

3.3 Abruf der WSDL

Nun kann man daran gehen, sich mit dem eigentlichen Aufruf des Services zu beschäftigen. Wie im Abschnitt 1.3 beschrieben wurde, stellt jeder SOAP-Service dazu das sogenannte WSDL-Dokument zur Verfügung. Es lässt sich ganz einfach im Browser ansehen, indem man an den URI des Services den Parameter `?wsdl` anhängt. In unserem Fall lautet diese URI:

```
http://localhost:4204/psn-ejb/DomainManagerBean?wsdl
```

Damit wird das XML-Dokument aus Abbildung 2 angezeigt. Wie dort schon besprochen, ist dieses Dokument für menschliche Leser nicht intuitiv verständlich. Leider kann auch SAS nicht direkt damit arbeiten. Es gab in der Version 9.2 eine Option `xmltype=wsdl` in LIBNAME, aber diese ist in den folgenden Versionen nicht weiter gepflegt worden. Damit wäre ein eleganter Aufruf des Webservices über diese zwei Zeilen möglich gewesen.

```
filename wsdl url "http://localhost:4204/psn-ejb/PSNManager?WSDL";
libname psn xml92 xmltype=wsdl;
```

Um an dieser Stelle weiter zu kommen, empfiehlt sich der Einsatz eines externen Tools. Eine gute Wahl auf Grund seiner einfachen Benutzbarkeit ist *soapUI* (http://www.soapui.org). Der primäre Einsatzzweck dieses in JAVA geschriebenen Open-Source-Tools ist der Test von Webservices. Hier hilft es, die WSDL zu verstehen und daraus beispielhafte Requests und Responses zu generieren. Dazu lädt man von der Projektwebseite die aktuelle Version und entpackt das ZIP-File in einen beliebigen neuen Ordner auf der lokalen Festplatte. Vorausgesetzt, JAVA ist auf dem PC installiert, startet soapUI durch einen Doppelklick auf die Datei `bin/soapui.bat`.

Für die weitere Arbeit wird eigentlich nur eine der vielen Funktionen von soapUI benötigt. Man legt ein neues SOAP-Projekt an und trägt den URI des Servicegebers ein, um das WSDL-Dokument abzurufen. Dabei ist auf das Häkchen bei der Checkbox „Create Requests“ zu achten.

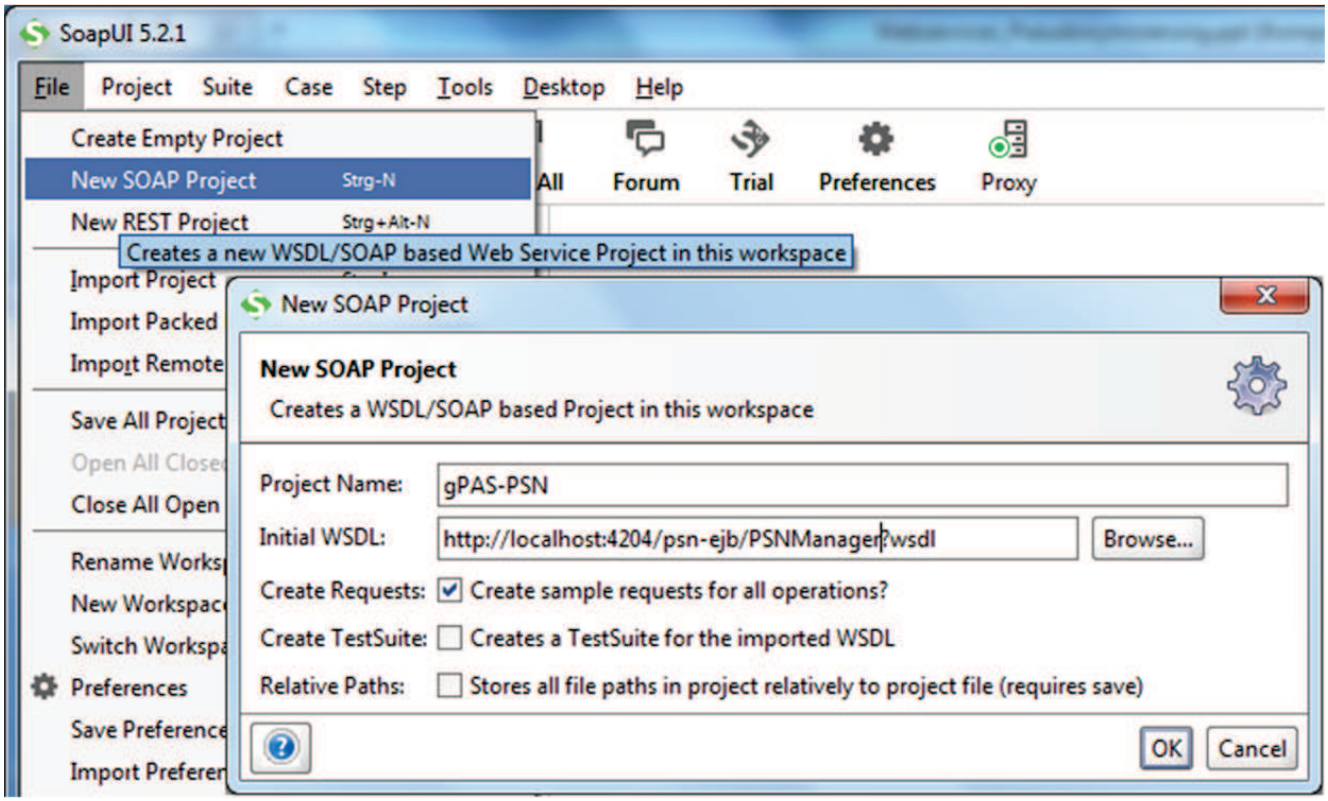

Abbildung 6: Abruf des WSDL-Dokuments mittels soapUI

3.4 Erzeugen des Requests

SoapUI interpretiert nun das WSDL-Dokument und stellt die Operationen des Services und ihre Parameter in einer übersichtlichen Baumstruktur dar, wie in der Abbildung 7 zu erkennen ist. Man sieht die im Abschnitt 2.2 beschriebenen Funktionen, von denen für unser Beispiel `getOrCreatePseudonymForList` genutzt werden soll. Der generierte Beispielrequest enthält die in Abbildung 3 gezeigten Elemente Envelope, Header und Body. Es gilt nun, mit den Mitteln von SAS zur XML-Verarbeitung einen solchen Request zu erzeugen.

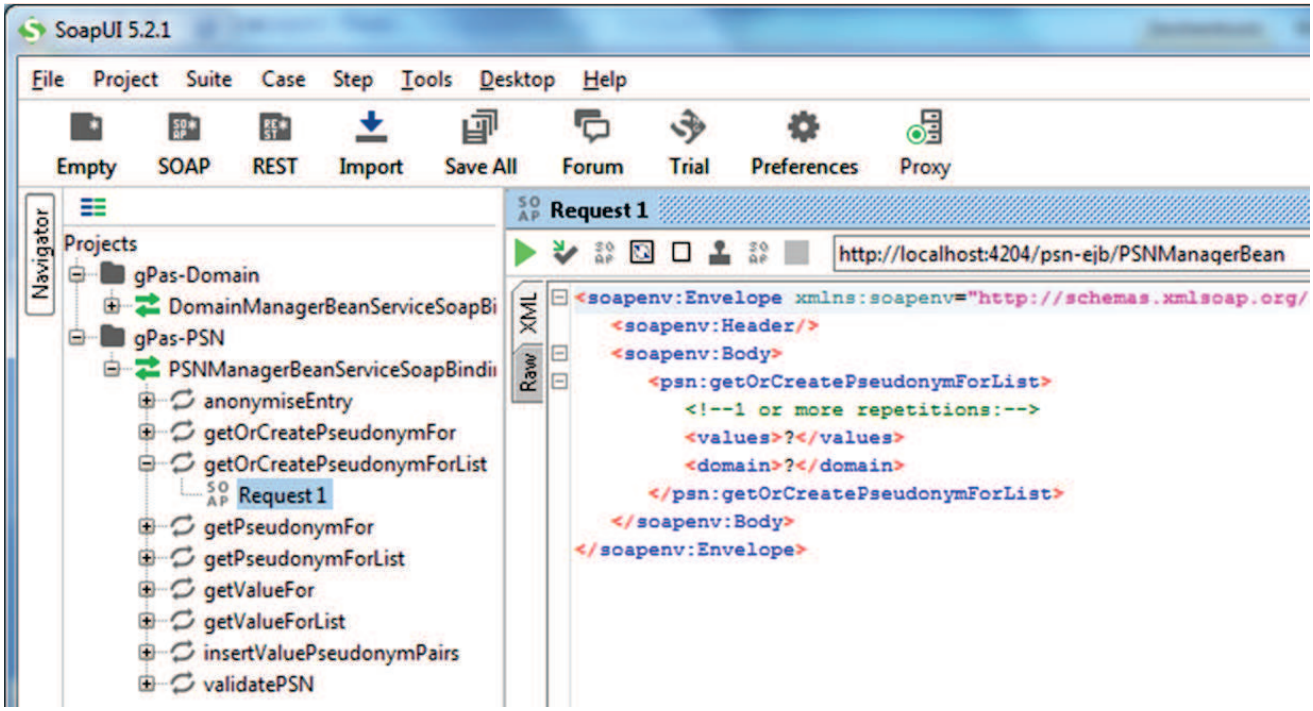

Abbildung 7: Beispielrequest für `getOrCreatePseudonymForList`

Dabei kommt uns SAS ein Stück weit entgegen. Sofern der Header keine Angaben enthält, wie es hier der Fall ist, braucht man sich um Envelope und Header nicht zu kümmern. Es muss nur der Inhalt von Body bereitgestellt werden. Dazu dient ein DATA-Step, der mittels PUT ein temporäres XML-File namens PSN_REQUEST erzeugt. Er schreibt zunächst das öffnende `<getOrCreatePseudonymForList>`-Element. Dann iteriert er über die Daten der Beispieltabelle SASHELP.BASEBALL und schreibt für jede Zeile den NAME als ein `<values>`-Element. Nach der letzten Zeile werden noch das `<domain>`-Element und das abschließende `<getOrCreatePseudonymForList>` ausgegeben.

```
*XML Request File definieren;
FILENAME PSN_REQUEST TEMP;

*Request File für Webservice erstellen;
data _null_;
  file PSN_REQUEST;
  set sashelp.baseball end=eof;
  if _n_=1 then do;
    put "<psn:getOrCreatePseudonymForList
              xmlns:psn='http://psn.ttp.ganimed.icmvc.emau.org/'>";
  end;
  put "  <values>" Name +(-1) "</values>";
  if eof then do ;
    put "  <domain>BaseBall-01</domain>";
    put "</psn:getOrCreatePseudonymForList>";
  end;
run;
```

Abbildung 8: DATA-Step zur Aufbereitung des Requests

3.5 Aufruf des Services

Nun kann der so erzeugte Request an den Service geschickt werden. Dazu stellt SAS mit PROC SOAP eine sehr elegante und einfache Methode zur Verfügung. Diese braucht im einfachsten Fall nur 3 Parameter: das Requestfile, einen Filenamen zum Abspeichern des Responses und den URI des Webservices.

```
*Talk to the Webservice;
filename PSN_RESPONSE "/folders/myfolders/soap-demo/PSN_liste.xml";
proc soap in=PSN_REQUEST
          out=PSN_RESPONSE
          url="http://localhost:4204/psn-ejb/PSNManager";
run;
```

PROC SOAP bietet noch eine Vielzahl mehr an Argumenten, die ausführlich in der Dokumentation beschrieben sind [8]. So könnte man Username und Passwort übergeben, falls der Service eine Authentifizierung verlangt oder den Request über einen Proxy routen. Sehr hilfreich in der Entwicklungsphase kann das Argument `DEBUG` sein. Hier kann man einen Dateinamen angeben, in den PROC SOAP ein ausführliches Protokoll der Kommunikation mit dem Servicegeber schreibt. Die Ausschriften im Logfile sind ansonsten etwas sparsam und beschränken sich auf die verbrauchte Zeit (real time und CPU time).

Wenn alles geklappt hat, kann man nun mit einem Editor in den erhaltenen Request schauen und das Ergebnis des Serviceaufrufes überprüfen. Im Beispiel sieht man, dass jedem Namen eines Baseballspielers im `<key>`-Element ein Pseudonym der gewünschten Form im `<value>`-Element zugeordnet wurde.

```
<?xml version="1.0"?>
- <ns2:getOrCreatePseudonymForListResponse xmlns:ns2="http://psn.ttp.ganimed.icmvc.emau.org/">
  - <return>
    - <map>
      - <entry>
          <key xsi:type="xs:string" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">Harrah, Toby</key>
          <value xsi:type="xs:string" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">pl_470109</value>
        </entry>
      - <entry>
          <key xsi:type="xs:string" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">Brooks, Hubie</key>
          <value xsi:type="xs:string" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">pl_925210</value>
        </entry>
```

Abbildung 9: Das erhaltene Response-Dokument

3.6 Einlesen des Responses

Nun gilt es, die Daten aus dem XML-Response für die weitere Verarbeitung in SAS verfügbar zu machen. Dazu bietet sich eine XML-Map an, in der wir eine Tabelle definieren, deren Variable als XPath-Ausdrücke zu den Elementen `<key>` und `<value>` definiert sind.

```
filename MAP "/folders/myfolders/soap-demo/psn_result.map";
data _null_;
  file MAP;
  put "<?xml version='1.0' encoding='windows-1252'?>";
  put "<SXLEMAP name='AUTO_GEN' version='2.1'>";
  put "  <NAMESPACES count='1'>";
  put "    <NS id='1,prefix='ns2'>http://psn.ttp.ganimed.icmvc.emau.org/</NS>";
  put "  </NAMESPACES>";
  put "  <TABLE description='gPAS' name='gPAS'>";
  put "    <TABLE-PATH syntax='XPathENR'>
             /{1}getOrCreatePseudonymForListResponse/return/map/entry</TABLE-PATH>";
  put "      <COLUMN name='key'>";
  put "        <PATH syntax='XPathENR'>
            /{1}getOrCreatePseudonymForListResponse/return/map/entry/key</PATH>";
  put "        <TYPE>character</TYPE>";
  put "        <DATATYPE>string</DATATYPE>";
  put "        <LENGTH>50</LENGTH>";
  put "      </COLUMN>";
  put "      <COLUMN name='value'>";   - - - -
  put "      </COLUMN>";
  put "  </TABLE>";
  put "</SXLEMAP>";
run;
```

Abbildung 9: Eine XML-Map zur Verarbeitung des Responses

Zum Einlesen der Ergebnisdaten des Services definiert man nun eine SAS-Bibliothek auf Basis des XML-Requests und der XML-Map und hat im Ergebnis eine Tabelle mit der Zuordnung der Pseudonyme zu den Namen der Baseballspieler.

```
libname PSN_list xmlv2 xmlfileref=PSN_RESPONSE xmlmap=PSN_MAP;
```

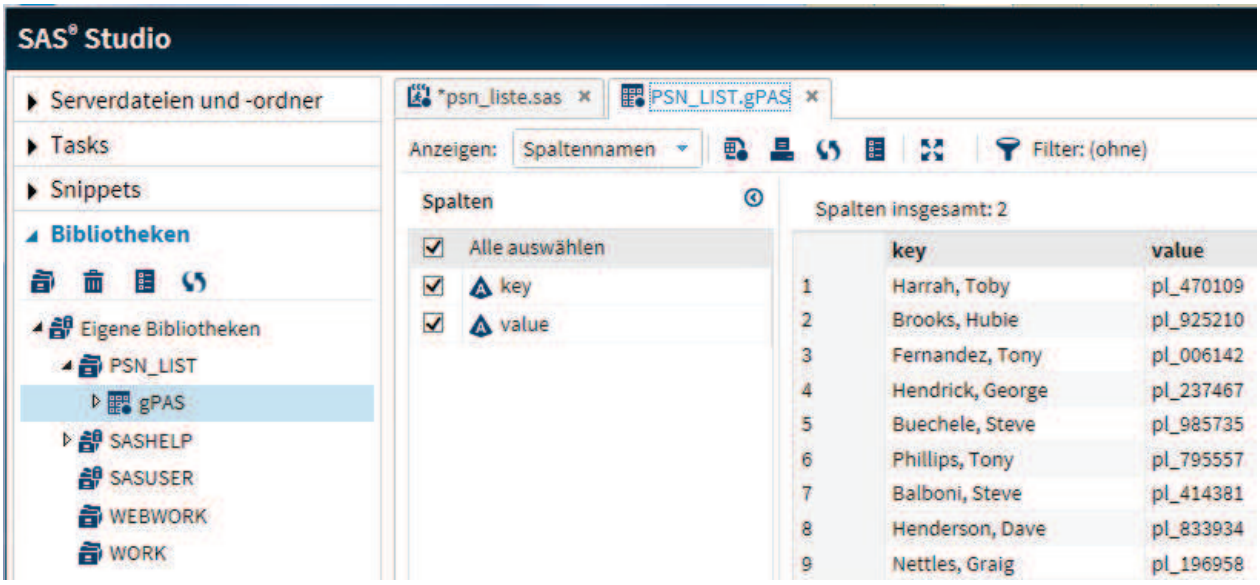

Abbildung 10: SAS-Tabelle mit den zugeordneten Pseudonymen

In einem abschließenden DATA-Step kann nun die originale Tabelle SASHELP.BASEBALL mit der Tabelle der Pseudonyme verknüpft werden, um den gewünschten Datensatz zur Weitergabe an die Analyse zu erstellen.

4 Zusammenfassung

Die folgende Grafik verdeutlicht das beschriebene Vorgehen noch einmal im Überblick. SAS unterstützt mit PROC SOAP die Nutzung von Webservices prinzipiell sehr gut, allerdings kommt man nicht um die intensive Beschäftigung mit der XML-Verarbeitung bei der Aufbereitung des Requests und dem Einlesen des Responses herum. Zur Vorbereitung des Zugriffs auf einen Webservice helfen externe Tools wie soapUI, die den Anwender bei der Interpretation der formalen Servicebeschreibung in WSDL unterstützen. Hat man geschilderten Prozess einmal erfolgreich etabliert, bietet es sich natürlich an, daraus ein wiederverwendbares Makro zu machen, so dass eine Funktion wie die beschriebene Pseudonymisierung einfach in weitere Prozesse des Datenmanagements eingebunden werden kann.

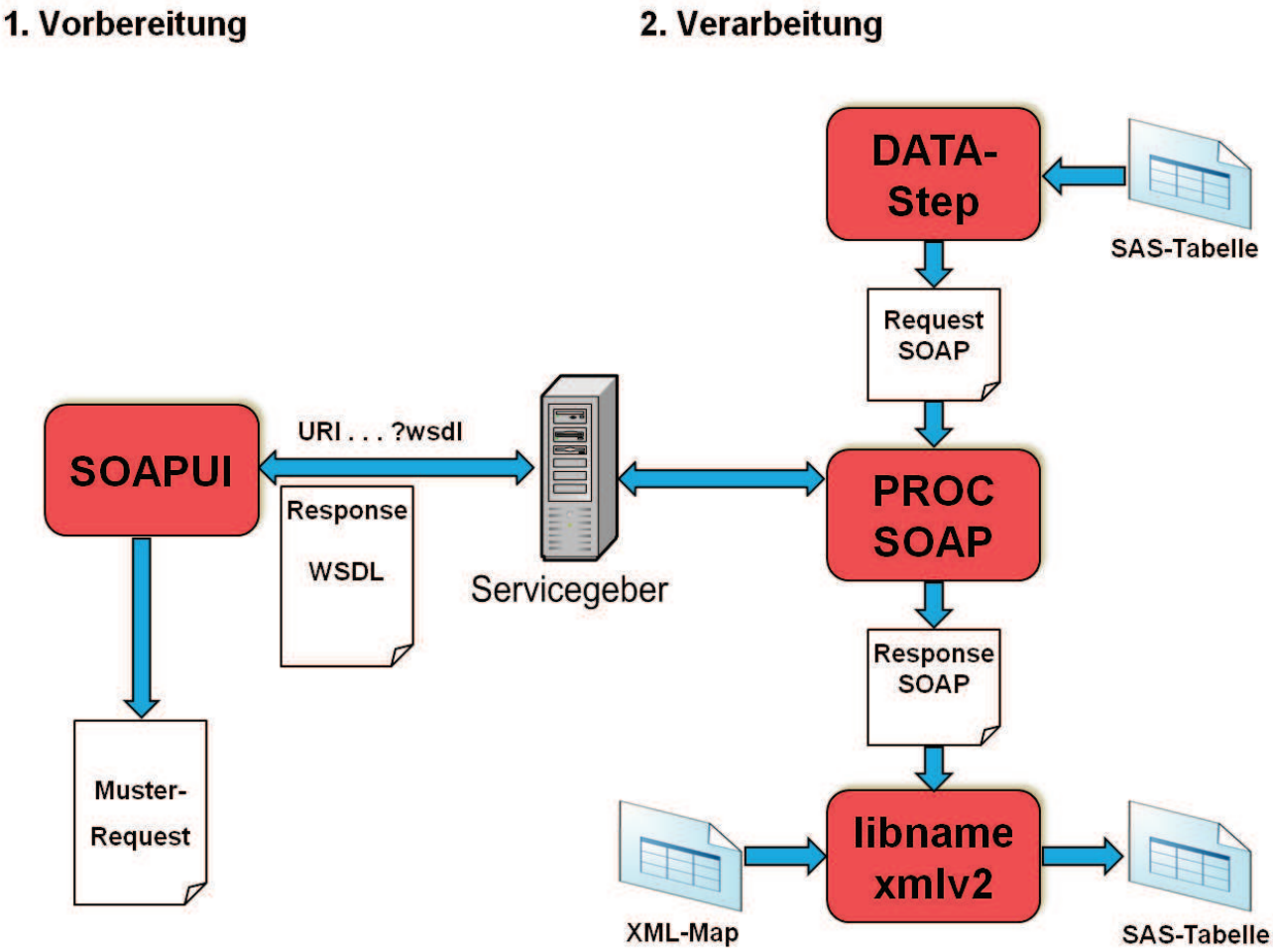

Abbildung 11: Zusammenfassung

Literatur

[1] G. Starke: Effektive Software-Architekturen. Carl Hanser Verlag München, 4. Auflage 2009 S. 331.

[2] K. Pommerening, J. Drepper, K. Helbing, T. Ganslandt: Leitfaden zum Datenschutz in medizinischen Forschungsprojekten. Generische Lösungen der TMF 2.0. Medizinisch Wissenschaftlichen Verlagsgesellschaft Berlin 2014.

[3] http://nako.de

[4] http://www.medizin.uni-greifswald.de/gani_med/

[5] http://web1-zkkr.zkkr.med.uni-greifswald.de/

[6] http://mosaic-greifswald.de

[7] M. Bialke, T. Bahls, C. Havemann, J. Piegsa, K. Weitmann, T. Wegner, et al.: MOSAIC. A modular approach to data management in epidemiological studies. Methods of Information in Medicine. 2015; 54(4):364-371.

[8] SAS Institute Inc.: Base SAS(R) 9.4 Procedures Guide – SOAP Procedure. http://support.sas.com/documentation/cdl/en/proc/68954/HTML/default/viewer.htm#n0vft359cr5yyon15n7b6zvrv102.htm