TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

DIPLOMARBEIT

User-Interface Generator

für SemCrypt Applikationen

Ausgeführt am Institut für

Informationssysteme

der Technischen Universität Wien

unter der Anleitung von ao.Univ.Prof. Dipl.-Inf. Dr.Ing. Dorn

durch

Sigfried Schweigl
Hundsheim 50
3512 Mautern

Wien, im Dezember 2006

Danksagung

Hiermit möchte ich mich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben, vor allem aber:

- Bei meinen Betreuern ao.Univ.Prof. Dipl.-Inf. Dr.Ing. Dorn Jürgen und Mag. Dipl.Ing. Wolfgang Schreiner für Beratungen, Unterstützungen und Hilfestellungen jeglicher Art. Sie waren jederzeit für Fragen offen und gaben mir durch ihr konstruktives Feedback richtungsweisende Hinweise.

- Bei meinen Eltern für die Unterstützung in all den letzten Jahren. Sie stellten dabei ihre eigenen Bedürfnisse in den Hintergrund, um mir mein Studium so sorgenfrei wie möglich zu gestalten.

- Bei meiner Freundin Daniela, die meine Launen, des Studiums wegen, ertragen und mich immer wieder aufgebaut hat.

Danke!

Abstract

Die vorliegende Diplomarbeit beschäftigt sich mit der Erstellung eines Systems, das die automatische Generierung eines Web-Interfaces aus XML-Eingabedaten mit einem fix vorgegebenen Schema ermöglicht. Für die Generierung werden daher verschiedene Code-Generator Modelle vorgestellt. Weiters werden einige geräteunabhängige Markup-Sprachen beleuchtet, wodurch es ermöglicht werden soll, dass das generierte Web-Interface auf verschiedensten Geräten wie Handy, PDA, etc. dargestellt werden kann und in weiterer Folge Datenbankabfragen (XPath/XUpdate-Anweisungen) auf eine XML-Datenbank erlaubt. Der Generator mit seiner Bedienoberfläche wird als Eclipse Plug-in realisiert. Da diese Arbeit ein Beitrag zur Unterstützung für Entwickler von SemCrypt Anwendungen ist, wird das Forschungsprojekt SemCrypt ebenfalls kurz erläutert.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit ist ein Arbeiten ohne die Unterstützung durch Computer nicht mehr vorstellbar. Durch diese weite Verbreitung und die weltweite Vernetzung von Computersystemen kommen vor allem auf Systemadministratoren immer neue Anforderungen, speziell in der Datensicherheit, zu. Früher verwahrten verschließbare Schränke wichtige Daten, heute sind Server mit Datenbanken dafür verantwortlich. Vor allem für Unternehmen ist es wichtig, dass Daten geheim gehalten werden. Das wären unter anderem Daten, die den Wettbewerbsvorteil gegenüber ihren Mitbewerbern sichern oder sogar Neuentwicklungen, die oft nur unter großen finanziellen Aufwand zustande kamen. Da es Menschen gibt, die sich Profit aus dem Erlangen von fremden Informationen erhoffen, bzw. deren Ziel die mutwillige Zerstörung von Daten jeglicher Art ist, bedarf es Schutzvorrichtungen, um diese Personen davon abzuhalten. Da der „Feind“ aber nicht immer nur von außen kommt, müssen außerdem verschiedene Zugriffsberechtigungen im eigenen Unternehmen vergeben werden. So soll ein Mitarbeiter nicht die gleichen Berechtigungen haben wie ein Vorstandsmitglied.

Weiters ist es heutzutage notwendig Daten überall abrufen zu können. Oft kann es aber sein, dass man seinen Laptop nicht dabei hat und so mittels anderen mobilen Geräten wie Handy, PDA, etc. Zugang sucht. Sei dies nur, um eine Fahrplanauskunft via Handy zu erfragen. Dies stellt aber viele Entwickler vor ein Problem. Wie kann man Daten aufbereiten, damit sie auf möglichst vielen Endgeräten darstellbar sind. HTML bzw. XHTML ist hier bei weitem nicht einsetzbar und wird hauptsächlich für Browser verwendet, für Handys verwendet man zurzeit WML. XHTML-Basic, welches nur einen Teil aus XHTML zur Verfügung stellt, würde eine Darstellung auf Handys und Browser gleichermaßen ermöglichen, bietet aber wegen der geringen Unterstützung verschiedenster Tags und Formen kaum Komfort und kann deshalb nicht die Lösung sein.

Auf diesen Überlegungen setzt das Forschungsprojekt SemCrypt an, dessen Ziel es ist, Queries und Updates auf verschlüsselten XML Dokumenten durchzuführen. Somit können sich die XML-Daten auf Servermaschinen befinden, auf welche in der Regel kein Einfluss genommen werden kann (untrusted server), da die Verschlüsselung und Entschlüsselung der XML-Daten auf der Client-Seite erfolgt und somit server-seitig weder die Dokumentenstruktur noch der Dokumenteninhalt bekannt gegeben wird. Es ist somit möglich Daten sicher auf beliebigen Fremdservern zu speichern, deren Zugriffskontrolle man nicht sicher sein kann. Diese Daten sollen außerdem über ein Web-Interface, welches geräteunabhängig sein sollte, abrufbar bzw. veränderbar sein.

Diese Diplomarbeit beschäftigt sich mit einem Beitrag zur Unterstützung für Entwickler von SemCrypt Anwendungen im SemCrypt Projekt. Und zwar mit der Implementierung eines Generators, welcher für die Generierung des zuvor genannten Web-Interfaces verantwortlich ist. Der Generator wird mit einem Eclipse Plugin realisiert, wobei dieses Plugin einem Anwender (zum Beispiel Administrator) bei der Berechtigungsvergabe dienen soll, indem dieser je Anwenderrolle mögliche Datenbankabfragen in Form von XPath- bzw. XUpdate-Anweisungen erlauben oder verbieten kann. Daraus soll das Plugin ein Web-Interface generieren, welches

beliebigen Anwendern, abhängig vom Berechtigungsgrad, Datenbankabfragen gestattet. Das generierte GUI sollte weiters geräteunabhängig sein, so dass diverse Browser, Handys, PDAs, etc. darauf zugreifen können.

1.1 Übersicht

Nach der einleitenden Darstellung und der Vorstellung der Motivation für diese Arbeit werden in Kapitel zwei die Grundlagen der verwendeten Technik erläutert. Es wird hier neben wichtigen Begriffen wie XML, XUpdate, XPath oder Code Generierung auch das Projekt SemCrypt vorgestellt. Das nächste Kapitel beschäftigt sich mit dem Human Resource Management und seiner Bedeutung im SemCrypt Projekt. Im vierten Kapitel wird der Entwurf der Arbeit präsentiert. Dieser beinhaltet unter anderem die Vorstellung verschiedener Markup-Sprachen und eine Darstellung des UI-Aufbaus. Kapitel fünf zeigt eine Gesamtarchitektur und beschäftigt sich mit der Implementierung des Plug-ins. Hier wird der Generierungsvorgang näher beleuchtet, sowie die einzelnen Servlets, welche Verwendung finden. Die Servletarchitektur ist ebenfalls Bestandteil dieses Kapitels. Kapitel sechs zeigt das Endprodukt mit einigen Screenshots der Arbeit und im siebenten Kapitel wird eine Zusammenfassung mit Ausblick der Arbeit präsentiert. Die letzten Kapitel bilden die Literaturreferenzen und der Anhang.

1.2 Verwendete Ressourcen

Dieses Kapitel dient der Information über die eingesetzten Tools, der Testumgebung, sowie der verwendeten Technologien, welche für die Entwicklung des Prototyps eingesetzt wurden.

Testumgebung:

- Microsoft Windows XP (SP 2)

Eingesetzte Tools:

- Java 1.4.2_04
- Eclipse 3.1.0
- Tomcat 5.5
- Internet Explorer 6.0
  zugehörige XForms Plug-ins: Novell XForms Explorer, FormsPlayer 1.4.3.1033
- Mozilla Firefox 1.5
  zugehörige XForms Extension: Mozilla XForms 0.5
- TcpTrace v0.8.0.712 (für Traces zwischen Browser und Converter)
- Altova XMLSpy 2006 (für die Erstellung erster XForms Beispiele)
- Berkeley DB XML 2.2.13 (Testdatenbank)

Benutzte Technologien:

- XML, XForms, XPath, XSL, XUpdate, XML-Schema

2 Grundlagen

Dieses Kapitel enthält die wichtigsten Grundlagen der im Dokument vorkommenden Technologien und beschreibt diese kurz.

2.1 Semistrukturierte Daten

2.1.1 XML

Die Extensible Markup Language (erweiterbare Auszeichnungssprache) ist eine Teilmenge von SGML. SGML [27] ist eine Metasprache mit deren Hilfe man verschiedene Auszeichnungssprachen für Dokumente definieren kann. XML ist ebenso eine Metasprache, also eine Sprache um andere Sprachen zu definieren. Ein XML Dokument ist in zwei Teile gegliedert. Der erste Teil ist der Header, welcher Informationen wie Versionsnummer, Namensräume etc. beinhaltet. Dieser dient der Bearbeitung des Dokumentes für Parser oder Prozessoren. Der andere Teil beinhaltet den eigentlichen Inhalt des Dokumentes in einer baumförmigen Struktur. Dadurch dass XML-Dokumente eine textbasierende Struktur aufweisen, sind sie einfach portierbar und lesbar. Durch eben genannte Eigenschaften findet XML weltweit Verwendung. [28] Anschließend ein XML-Beispiel aus dem HR-Bereich für PersonName:

```
<PersonName>
  <FormattedName>Max Mustermann</FormattedName>
  <LegalName>Dr. Max Mustermann</LegalName>
  <GivenName>Max</GivenName>
  <FamilyName>Mustermann</FamilyName>
  <Affix type="qualification">Dr.</Affix>
  <Affix type="formOfAddress">Herrn</Affix>
</PersonName>
```

Zum Abschluss sei noch zu erwähnen, dass in XML-Dokumenten nicht jedes Zeichen darstellbar ist. Man denke dabei an Zeichen wie das „=“. Hier greift man auf Unicode-Zeichen zurück. Einzelne Unicode-Zeichen können durch &#n; bzw. &#xh; referenziert werden (n=Unicode-Zeichencode dezimal, h=Unicode-Zeichencode hexadezimal). Die einzigen Ausnahmen bilden folgende Zeichen:

- < <
- > >
- & &
- " "
- ' '

Diese können durch ihre vordefinierten Entitäten dargestellt werden.

2.1.2 XML-Schema

Ein XML-Schema ist eine Empfehlung von W3C und soll XML-Dokumentstrukturen definieren. Zusätzlich ist die Vergabe von Datentypen möglich. Somit ist es möglich ein XML-Dokument in seiner Struktur komplett zu definieren. Anschließend lässt sich das geschriebene Dokument gegen das Schema validieren. Ein Schema ist ebenfalls ein XML-Dokument und erlaubt komplexere Zusammenhänge als mit einer DTD zu beschreiben, was aber natürlich seinen Preis im Erlernen hat. Ein Schema-Auszug für vorher gezeigtes PersonName Beispiel:

```
<xsd:complexType name="PersonNameType">
  <xsd:sequence>
    <xsd:element name="FormattedName" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            ...
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="LegalName" type="xsd:string" minOccurs="0"/>
    <xsd:element name="GivenName" type="xsd:string" minOccurs="0"/>
    ...
```

2.1.3 XSLT

XSLT ergibt sich aus XSL Transformation, wobei XSL Extensible Stylesheet Language bedeutet. Wie der Name Transformation schon sagt, ist XSLT eine Transformation von XML Dokumenten. Die Transformation geschieht anhand von festgelegten Regeln und führt als Ergebnis einen Quellbaum in einen Ergebnisbaum über (siehe Abbildung 1). Die Regeln werden in einem Stylesheet verfasst, wobei eine Regel aus zwei Komponenten besteht: Dem Muster, welches gegen die Knoten im Dokument geparst wird und einem Template, um den Zielbaum zu formen. [9]

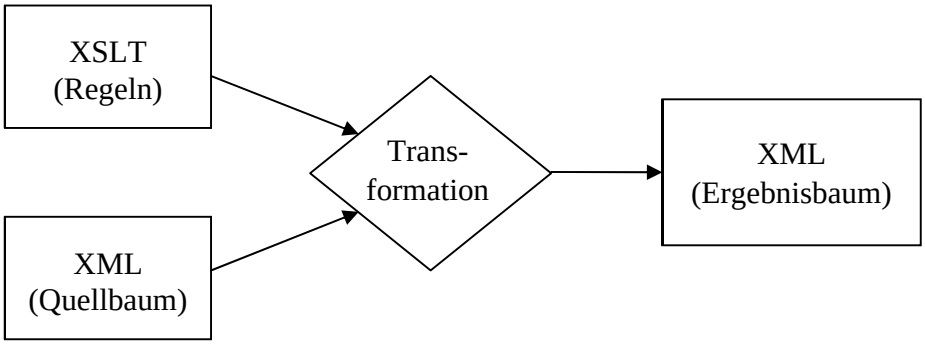

Abbildung 1: Ablauf einer Transformation

Eine Transformation geschieht mit XSLT Prozessoren, wobei bei den gängigsten Browsern diese Prozessoren bereits inkludiert sind, wie zum Beispiel beim Internet Explorer oder Mozilla Firefox. Ein einfaches Beispiel zeigt folgender Auszug einer XSLT-Datei, angewandt auf das PersonName XML-Beispiel von vorher. Hier wird der LegalName und FamilyName aus dem PersonName Block gelesen und in eine HTML-Tabelle transformiert:

```
<xsl:template match="xml">
  <html>
    <head>
      <title>PersonName</title>
    </head>
    <body>
      <table border="1">
        <tr bgcolor="#9acd32">
          <th align="left">Name</th>
          <th align="left">FamilyName</th>
        </tr>
        <xsl:for-each select="/xml/PersonName">
          <tr>
            <td><xsl:value-of select="LegalName"/></td>
            <td><xsl:value-of select="FamilyName"/></td>
          </tr>
        </xsl:for-each>
      </table>
    </body>
  </html>
</xsl:template>
```

2.1.4 XPath

Die Hauptaufgabe von XPath liegt in der Adressierung von Teilen eines XML-Dokumentes. Dies ist zum Beispiel mit einem Dateisystem unter Unix oder DOS vergleichbar, nur liegt hier nicht ein Dateisystem zu Grunde, sondern ein XML-Dokument, wobei XPath-Ausdrücke auf XML-Knoten (Nodes) referenzieren. Wie auch im Dateisystem gibt es hier absolute als auch relative Pfadangaben. Auf das PersonName XML-Beispiel Bezug nehmend, würde der LegalName wie folgt adressiert werden: /PersonName/LegalName. Auch Attribute lassen sich einfachst mit einem führenden „@“ adressieren, wie anhand des Attributes „type“ gezeigt werden kann: „/PersonName/Affix/@type“. Zusätzlich werden noch einfachste Funktionalitäten wie Manipulationen von Zeichenketten, booleschen Werten und Zahlen unterstützt. Solche Funktionen wären unter anderem „string()“ und „number()“. Für vorher gezeigtes LegalName-XPath-Beispiel würde die Funktion „string(.)“ den Wert „Dr. Max Mustermann“ liefern, wobei der „.“ den aktuellen Knoten angibt. Hauptanwendung findet XPath derzeit in XSL. Die aktuelle Version von XPath ist Version 1.0 vom 16. November 1999. [10]

2.1.5 XUpdate

XUpdate ist eine Sprache um XML Dokumente zu aktualisieren. Es bedient sich dabei vielfach der XPath Ausdrücke. [50] Im Nachfolgenden ein kurzer Überblick über die möglichen XUpdate Anweisungen.

XUpdate unterstützt folgende Modifikationsmöglichkeiten:

- xupdate:insert-before, xupdate:insert-after

  Mit der “insert” Anweisung lassen sich Elemente, Attribute, Texte, Processing Instructions und Kommentare in einem XML Baum einfügen. Wo genau im Baum eingefügt werden soll, wird durch das „select“ Attribut angegeben. Jetzt entscheidet noch das „before“ oder „after“, ob an der selektierten Stelle vorher oder nachher eingefügt werden soll. Anschließend ein Beispiel einer „xupdate:insert-before“ Anweisung, bei welcher vor dem Namen (FormattedName) eine UserId eingefügt wird:

```
<xupdate:modifications version="1.0"
                          xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:insert-before select="/PersonName/FormattedName">
    <xupdate:element name="UserId">muster</xupdate:element>
  </xupdate:insert-before>
</xupdate:modifications>
```

- xupdate:append

  Mit „append“ kann ein Element, ein Attribut, ein Textelement, eine Processing Instruction oder ein Kommentar erzeugt bzw. an einer Stelle im Baum angehängt werden. Auch hier muss wieder mit einem „select“ Attribut die Stelle im Baum angegeben werden, wo besagte Anweisung angehängt werden soll. Im Anschluss ein Beispiel einer „append“ Anweisung, in welcher eine Adresse angehängt wird:

```
<xupdate:modifications version="1.0"
                          xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:append select="/PersonName/Affix">
    <xupdate:element name="Address">Musterhausen</xupdate:element>
  </xupdate:append>
</xupdate:modifications>
```

- xupdate:update

  Diese Anweisung aktualisiert den Inhalt des mit dem „select“ Attribut angegebenen Knotens. So kurz wie die Beschreibung, so einfach auch das folgende Beispiel, welches den FamilyName des im vorher gezeigten „insert“ Beispiels von Mustermann auf Musterfrau ändert.

```
<xupdate:modifications version="1.0"
                          xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:update select="/PersonName/FamilyName">
    Musterfrau
  </xupdate:update>
</xupdate:modifications>
```

- xupdate:remove

  Bei dieser Anweisung wird ein selektierter Knoten aus dem XML Baum gelöscht. Hier ein Beispiel, welches die bei Append hinzugefügte Adresse wieder löscht:

```
<xupdate:modifications version="1.0"
                          xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:remove select="/PersonName/Address"/>
</xupdate:modifications>
```

- xupdate:rename, xupdate:variable, xupdate:value-of, xupdate:if

  Die genannten Anweisungen finden weniger Anwendung, werden hier aber der Vollständigkeit wegen, ebenfalls aufgezeigt. Mit „rename“ kann ein Attribut oder Element umbenannt werden. Mit „variable“ kann man einen Namen zu einem Wert binden und „value-of“ kann diese angelegte Variable erfragen. Die „if“ Anweisung ermöglicht bedingte Verarbeitungen.

2.1.6 XAccess

Für das SemCrypt Projekt wurde ein Zugriffsmechanismus entwickelt, genannt XAccess, der es gestattet, eine große Anzahl von Benutzergruppen zu definieren. Anhand dieser Benutzergruppen soll es dann möglich sein, Benutzerinformationen zu speichern, die bestimmen wer welche Daten lesen oder verändern darf. Dies geschieht in einem definierten XML-Schema, welches Elemente beschreibt, die für die Zugriffskontrolle benötigt werden. Ein valides Dokument besteht daher aus einer Liste von Rollen (z. B. Gruppenleiter, Projektleiter, Mitarbeiter, …), aus allen möglichen XPath-Pfaden (beinhalten auch die Attribute), die in einem zu überwachenden XML-Dokument vorkommen und einer Liste aus Berechtigungen, die definieren wer auf welche Daten (XPath) zugreifen (lesen, schreiben, …) darf. XPath-Pfade werden folgendermaßen angegeben:

- (“/” “@”? (<prefix> ”:”)? <value>)*

Alle Pfade werden absolut angegeben Handelt es sich um ein Attribut, so wird ein „@“ vorangestellt. Der <Prefix> bezeichnet den Namespace-Prefix und der <Value> den Namen des Elements oder des Attributs. Anschließend ein Auszug des definierten Schemas, welches die verwendeten Zugriffsmechanismen für die Arbeit zeigt:

```
<XAccess>
  <SecurityRoles>
    <SecurityRole>
      <RoleName>string</RoleName>1
      ...
    </SecurityRole>+
  </SecurityRoles>1
  <SecurityPaths>
    ...
    <SecurityPath>
      <PathValue>string</PathValue>1
      <Type>string</Type>1
      ...
    </SecurityPath>+
```

```
  </SecurityPaths>1
  <SecurityRights>
    ...
    <SecurityRight>
      <RightName>string</RightName>1
      ...
    </SecurityRight>+
  </SecurityRights>1
  <Permissions>
    <Permission>
      <PathValue>string</PathValue>1
      <RoleName>string</RoleName>+
      <RightName>string</RightName>+
      <Restrictions>
        <RecursionDepth>nonNegativeInteger</RecursionDepth>?
        <Path>
          <Operation>string</Operation>?
          <Value>string</Value>1
        </Path>*
      </Restrictions>?
    </Permission>+
  </Permissions>1
  <Denials>
    <Denial>
      ...
    </Denial>+
  </Denials>1
</XAccess>1
```

Das Schema definiert noch weitere Tags, die hier aber nicht aufgelistet werden. Einerseits liegt das daran, dass sie entweder für diese Arbeit nicht benötigt, oder erst für zukünftige Version angedacht werden und andererseits, dass derzeit die Datenbank diese nicht unterstützt [59]. [67]

2.2 Code Generierung

Unter der Generierung eines Codes versteht man eine Technik, bei der mit einem Programm ein Code generisch zusammengesetzt wird. Wobei ein Code ein Programmcode, aber auch eine Skriptsprache, Markup-Sprache oder ein beliebiger Text sein kann. Programme, mit welchen Code Generierungen durchgeführt werden, reichen von kleinen Skripts bis hin zu großen

Anwendungen, welche abstrakte Modelle der Business Logik in Applikationen umwandeln. Es gibt also zahlreiche Methoden um Code Generierung durchzuführen. [16]

Code Generierung ist heute wichtiger denn je. Sinnvoll eingesetzt erspart sie Zeit und Kosten. Man kann zwischen mehreren Generatortypen unterscheiden, die im Folgenden nun aufgeführt werden:

2.2.1 Code Munger

Diese Art von Generator liest Eingabedaten und formt diese in das gewünschte Ausgabeformat um. Der neu gebildete Code kann komplett oder teilweise vom Design des Generators abhängen. Ein Beispiel hierfür ist JavaDoc, welches Kommentare, die im Code eingebettet sind, als HTML Dokument generiert. Die typische Ablauffolge ist in Abbildung 2 dargestellt. [16]

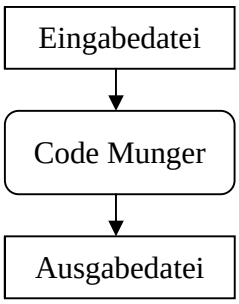

Abbildung 2: Ablauf eines Code Mungers [16]

2.2.2 Inline Code Expander

Bei einem Inline Code Expander wird eine neue Sprache entwickelt. Gewöhnlich wird aber eine existierende Sprache mit einigen Wörtern erweitert. Diese Erweiterung wird dazu verwendet, um in Vorlagen die Stellen zu markieren, wo der Generator dann den Code hineinplatziert. Der gesamte Ablauf ist in Abbildung 3 zu sehen. [16]

Hauptanwendung findet der Inline Code Expander im Einbetten von SQL Statements in bestehenden Code. Man benötigt ziemlich viel Infrastruktur für SQL, wie zum Beispiel die Datenbankverbindung, das Laufen lassen des Programms und einiges mehr. Nun ist das Ziel des Inline Code Expanders jenes, dass der Entwickler sich lediglich um die Syntax von SQL konzentrieren muss und alles andere, wie eben die Datenbankverbindung, dem Generator überlässt. [26]

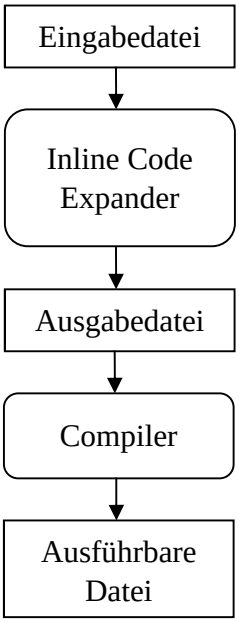

Abbildung 3: Ablauf eines Inline Code Expanders [16]

2.2.3 Mixed Code Generator

Hier wird die Eingabedatei mit neuem Inhalt überschrieben, somit gibt es keine Ausgabedatei in dem Sinn (siehe Abbildung 4). Es ist Aufgabe vom Generator ein Backup der Eingabedatei vor dem Manipulieren mit dem neuen Inhalt für den Fehlerfall zu haben. [26]

Es gibt hier ebenfalls Platzhaltervariablen, jedoch werden diese nicht ersetzt, sondern dienen dazu den Code an diesen Stellen anzuhängen. Dieser Typ ist dem Inline Code Expander Modell vorzuziehen, da er sich besser in IDEs als externes Tool integrieren lässt und die Debug-Möglichkeiten besser sind. Eine denkbare Einsatzmöglichkeit ist beispielsweise die Generierung von Programmlogik aus Kommentaren in der Eingabedatei, wobei aber die Kommentare erhalten bleiben sollen [16]

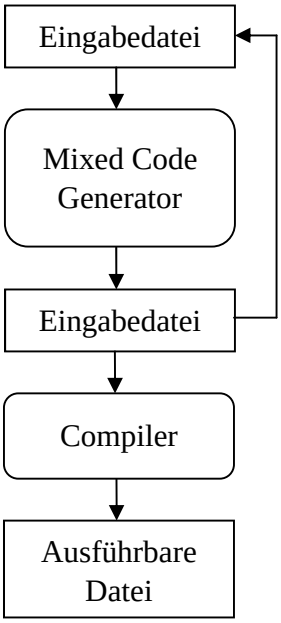

Abbildung 4: Ablauf eines Mixed Code Generators [16]

2.2.4 Partial Class Generator

Bei der Partial Class Generierung erzeugt ein Generator aus einem abstrakten Modell die gewünschten Daten. Die bisherigen Generatoren verwenden alle ausführbaren Code wie zum Beispiel C, Java oder SQL als Eingabedatei. Dieser Generator verwendet eine abstrakte Definition des Codes für die Eingabe. Anstatt wie die anderen Generatoren Code-Fragmente zu ersetzen bzw. herauszufiltern, generiert dieser den ganzen Code der Implementierung. Im Gegensatz zum Tier Generator Model muss der Partial Class Generator in Verbindung mit den angepassten Quelldaten verwendet werden um die Ausgabe zu überschreiben (siehe Abbildung 5). Dieser Generator ist eine Vorstufe auf dem Weg zu einem Tier Generator Modell. [26]

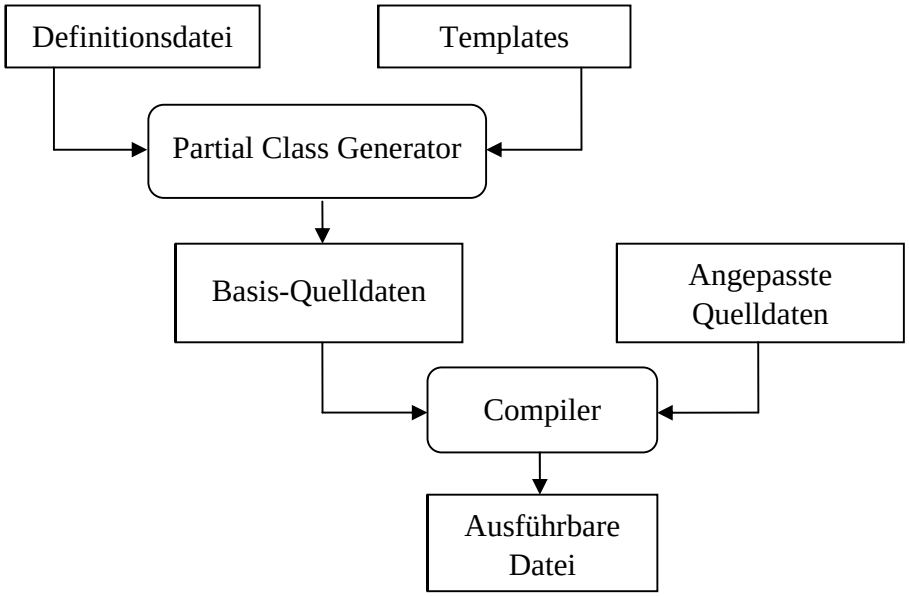

Abbildung 5: Ablauf eines Partial Class Generators [16]

2.2.5 Tier Generator Model

Ein Schichtgenerator ist in der Lage aus einem abstrakten Modell, welches normalerweise als XML-Struktur vorliegt, beliebig viele Schichten eines Systems zu erzeugen. Der Generator erhält aus der Definitionsdatei die Informationen um alle Klassen und Funktionen der Schicht zu kreieren. Mit diesen Informationen werden in Verbindung mit den Templates die Ausgabedaten erzeugt, welche bereit für die Integration in die Applikation sind (Abbildung 6). Im Unterschied zur Partial Class Generierung wird eine gesamte Anwendungsschicht erzeugt, die keine manuellen Eingriffe mehr benötigt. Der Entwicklungsaufwand für solche Generatoren ist deshalb deutlich höher. [26]

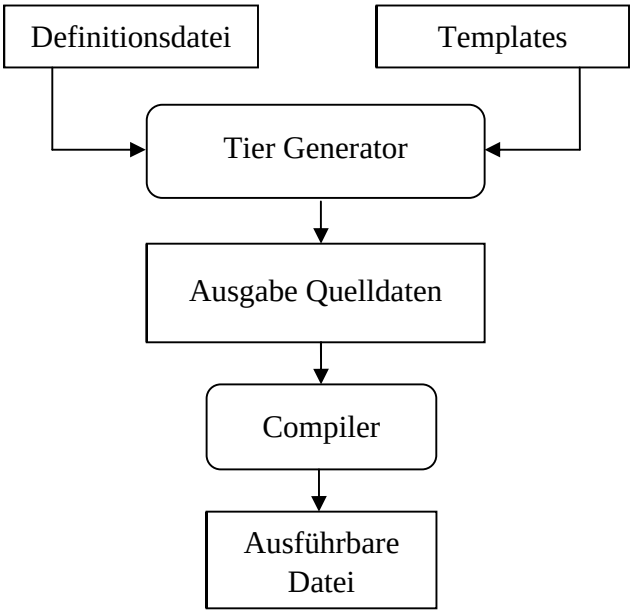

Abbildung 6: Ablauf eines Tier Generator Models [16]

2.3 SemCrypt

SemCrypt ist ein Forschungsprojekt und wird durch das FIT-IT Programm gefördert, welches eine Initiative des Ministeriums für Transport, Innovation und Technologie ist. Folgende Partner sind beteiligt: Johannes Kepler Universität Linz, Electronic Competence Center Vienna und EC3 NetworksGmbH. [61]

2.3.1 Idee des Projekts

Outsourcing ist heutzutage in aller Munde. Dieser Trend bewirkt, dass immer mehr Dokumente auf externen Servern gespeichert werden. Mit externen Servern ist das aber so eine Sache, man kann darauf vertrauen, dass die Dokumente geschützt sind, oder muss damit leben, dass der Inhalt eingesehen werden kann.

W3C entwickelte eine XML-Verschlüsselung mit der Voraussetzung, dass der Übertragungsweg sicher sein muss. Diese Technik erlaubt aber keine Speicherung von Daten, da weder Abfragen noch Aktualisierungen von Daten über verschlüsselte XML-Dokumente unterstützt werden. Aber gerade bei semantik-basierenden Dokument-Systemen ist diese Fähigkeit ein wesentlicher Bestandteil.

Und hier setzt nun das Forschungsprojekt an, mit der Idee Techniken für Abfragen und Aktualisierungen über verschlüsselte XML-Dokumente zu schaffen. Dies soll dadurch erreicht werden, indem das verschlüsselte Dokument auf den sicheren Client geladen und entschlüsselt wird. Somit kann man nun die Daten abfragen bzw. verändern. Anschließend wird das Dokument wieder verschlüsselt auf dem Server abgelegt. [61]

2.3.2 Anwendungsgebiet

Die ersten Einsatzmöglichkeiten eines Prototyps sollen im Bereich der Humanressourcen (HR) liegen. Dies ergibt sich aus etlichen Charakteristiken. Unter anderem sind dies: Umfassende Abfrage- und Update-Erfordernisse für HR-Informationen, hohe Sensibilität der behandelten Daten in HR-Teilbereichen, Umfang der zu erwarteten Daten erscheint XML geeignet, Spezifikationen im XML-Bereich sind bereits vorhanden (HR-XML) und werden weiterentwickelt, HR-Daten sind für den Austausch prädestiniert; XML = Austauschformat, uvm. Des Weiteren sollen neben dem HR-Bereich Anwendungen für eGovernment, eHealth, Tourismus, Versicherungswesen, dem allgemeinen Finanzbereich oder generell im Produktionsbereich beobachtet werden. [62]

2.3.3 Architektur

In Abbildung 7 folgt nun die Gesamtarchitektur von SemCrypt:

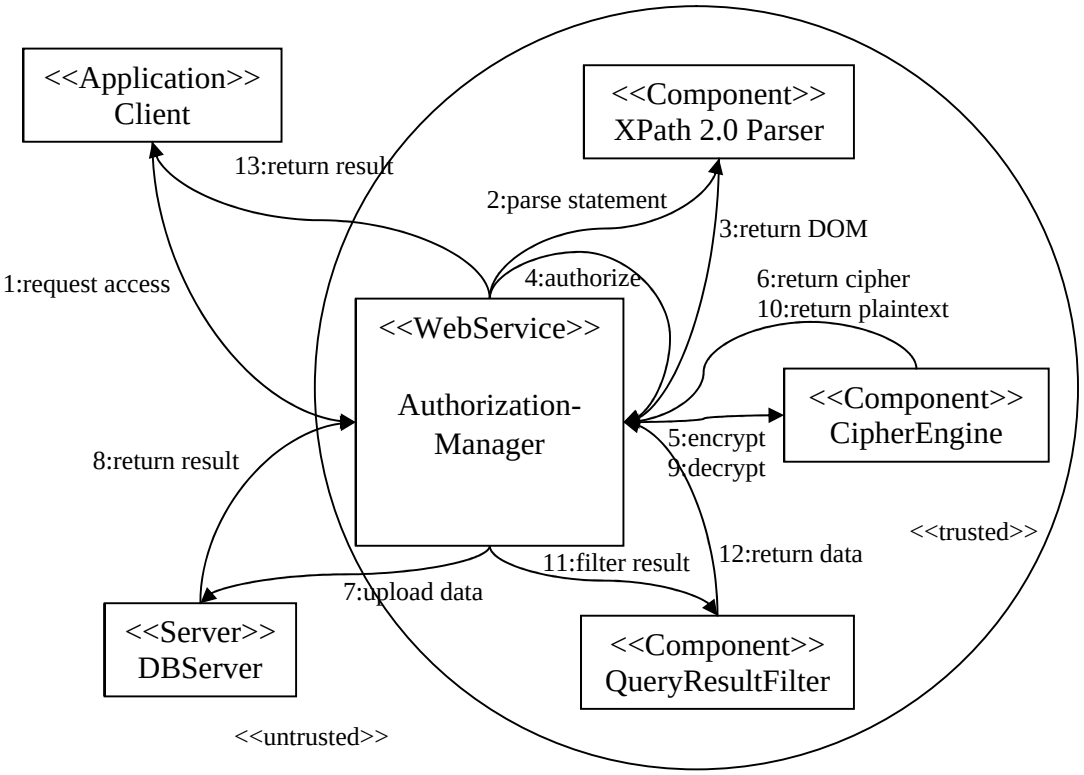

Abbildung 7: Architektur

Der Ablauf beginnt mit einem Anwender, welcher z. B. über einen Browser die Login-Page anfordert. Bei erfolgreicher Authentifizierung kann der Anwender eine Datenbankabfrage absenden (1) welche dann zum XPath Parser weitergeleitet wird (2). Dieser überprüft die Anweisung und liefert ein Java-Objekt, ähnlich DOM, zurück (3). Das DOM-Objekt wird dazu benutzt, um den Authentifizierungsprozess auszuführen (4). Die CipherEngine wird aufgerufen,

um Daten, welche neu eingefügt werden sollen, zu verschlüsseln (5,6), bzw. um Daten, die am Client angezeigt werden sollen, zu entschlüsseln (9,10). Geänderte Daten werden an den Datenbankserver geleitet und auf deren Ergebnis überprüft (7,8). Der QueryResultFilter-Aufruf für Validierungszwecke ist optional, liefert aber zusätzliche Sicherheit bezüglich sensibler Ergebnisdaten (11,12). Zuletzt wird das Ergebnis an den Client gesendet (13). [67]

2.3.4 Entwicklungs- und Laufzeitsystem

Für die Entwicklung von SemCrypt Komponenten wird Eclipse verwendet, welches die Installation von Java als Grundvoraussetzung auf dem System benötigt. Eclipse ist ein Open-Source Framework, welches häufig als Entwicklungsumgebung verwendet wird. Durch die scheinbare Plattformunabhängigkeit (SWT ist auf zahlreichen Plattformen erhältlich) kann es als Entwicklungssystem auf verschiedensten Systemen wie Workstations mit Linux, HP-UX, Solaris, Windows und einigen mehr eingesetzt werden [3]. Als XML Testdatenbank kann Xindice [65] oder Berkeley XML DB [64] eingesetzt werden. Dabei ist zu beachten, dass Xindice nur XPath 1.0 unterstützt. Als Application Server dient Tomcat, welcher mit einem Servlet-Container für das Web-Interface ausgestattet sein muss.

Für das Laufzeitsystem wird für das Web-Interface ebenfalls ein Application Server mit einem Servlet-Container benötigt. Der Einsatz von Eclipse ist hier nicht mehr von Nöten. Die von der Johannes Kepler Universität in Linz entwickelte XML Datenbank kommt hier zum Einsatz. Diese ist für XPath 2.0 spezifiziert.

2.4 Web-Interface Systeme

2.4.1 XUL

XUL (XML User-Interface Language), von Mozilla entwickelt, ist eine auf XML basierende Auszeichnungssprache, welche eine Erstellung von plattformunabhängigen grafischen Benutzeroberflächen für beliebige Applikationen ermöglicht. Dabei sind Standards wie XPath, XSLT und DOM Funktionen verwendbar. Selbst die grafische Oberfläche von Mozilla ist mit XUL implementiert, was zeigt, wie leistungsstark diese Sprache eigentlich ist. XUL läuft aufgrund der Portierbarkeit unter allen Windows-Versionen, auf Macintosh, Unix und Linux Systemen. [30]

Durch den Grundgedanken, dass Mozilla als Entwicklungsframework sowie als Webbrowser verwendet werden soll, wurde XUL ursprünglich für den Open Source Webbrowser Mozilla (Netscape 6) im Rahmen des XPToolkit-Projekts erarbeitet. Ab Netscape 6.x wird XPFE (Cross Platform Front End), eine auf XUL basierende Entwicklungsumgebung, verwendet. XPFE setzt ähnlich wie XHTML auf drei Komponenten auf: XUL, Cascading Style Sheets (CSS) und JavaScript. Bei XHTML existiert statt der XUL-Komponente die HTML-Komponente. Für die Darstellung und das Layout ist nicht XUL sondern CSS verantwortlich ist, sowie JavaScript für Events und Benutzerinteraktionen. XUL wird zur Beschreibung der GUI-Komponenten (Widgets) verwendet, die notwendige Information für das zu verwendende Stylesheet wird entweder direkt oder als Referenz angegeben. JavaScript ist ebenfalls ein wesentlicher Bestandteil, mit welchem viele Funktionalitäten erweitert werden können. [29]

Abschließend noch ein Beispiel eines XUL-Windows mit dem Ergebnis in Abbildung 8:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin" type="text/css"?>
<!DOCTYPE window>
<window title="Hello xFly"
 xmlns:html="http://www.w3.org/1999/xhtml
 xmlns=http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul
 style="background-color: white;"
 width="300"
 height="215”
 onload="centerWindowOnScreen( )">
  <script type="application/x-javascript"
   src="chrome://global/content/dialogOverlay.js"/>
  <vbox align="left">
    <label style="font-weight: bold;" value="Hello, Welcome to the xFly"/>
    <image src="http://books.mozdev.org/xfly.gif"/>
    <button label="hello xFly" oncommand="alert('Hello World');"/>
  </vbox>
</window>
```

Abbildung 8: Laden des „xFly“ Beispiels im Mozilla Browser [31]

2.4.2 UIML

Es war schon immer das Ziel plattform- und geräteunabhängige Applikationen zu schreiben. Mit HTML war die Plattformunabhängigkeit durch den Einsatz verschiedenster Browser gelungen. Außerdem war diese Sprache auch für Nicht-Programmierer ideal geeignet. Das war der erste Ansatz für die Entwickler von UIML (User Interface Markup Language). Sie stellten sich damals die Frage, ob es möglich sei eine Sprache zu entwickeln, die ebenbürtig mit einer Programmiersprache und ihren grafischen Tools (wie zum Beispiel C mit X-Windows, C++ mit MFC oder Java mit AWT) sein kann, gleichzeitig aber wie HTML auch für Benutzer ohne viel Programmierkenntnisse zu bedienen ist. Drei weitere Gründe beeinflussten die Entwicklung von UIML. Der erste war, dass XML Standard für deklarative Sprachen wurde. Ein weiterer war der Trend in Richtung drahtloser Technologien. Es war ein Umbruch, weg vom PC als Desktop, festzustellen. Dies zeigte sich auch an den rasant zunehmenden mobilen Geräten wie zum Beispiel PDA und Handy. UIML sollte somit auch für Geräte erweiterbar sein, die noch nicht erfunden wurden. Als letztes war noch die Einführung von CSS maßgeblich, welche mit ihrer Sprachbeschreibung unabhängig von den Zielanwendungen war.

Das Ergebnis war UIML1 im Jahr 1998, entwickelt von Harmonia. Mit dieser Entwicklung bemerkte man etliche Fehler, was schließlich 1999 zu UIML2 führte. Bei dieser Version wurde CSS wieder entfernt, da das notwendige Mapping zu komplex wurde. Die Darstellung erfolgt daher durch das <presentation> Element, welches das Mapping angibt. Mit diesem Mapping wird angegeben, auf welche Zielsprache UIML abgebildet werden muss (siehe dazu das „Hello World“ Beispiel weiter unten). Die aktuelle Version ist nun bei UIML3.1 vom März 2004. [33]

Um die Plattform- und Geräteunabhängigkeit zu erreichen, wurden in UIML einige Renderer entwickelt, durch welche eine UIML-Datei in eine Zielsprache konvertiert und interpretiert wird. Momentan unterstützte Zielsprachen sind Java, HTML, WML und VoiceXML. [29]

Die Struktur von UIML ist in Abbildung 9 dargestellt. UIML beschreibt die Benutzeroberflächen abstrakt, daher eine Aufteilung in sechs Bereiche: Struktur, Style, Inhalt, Verhalten, Präsentation und Logik.

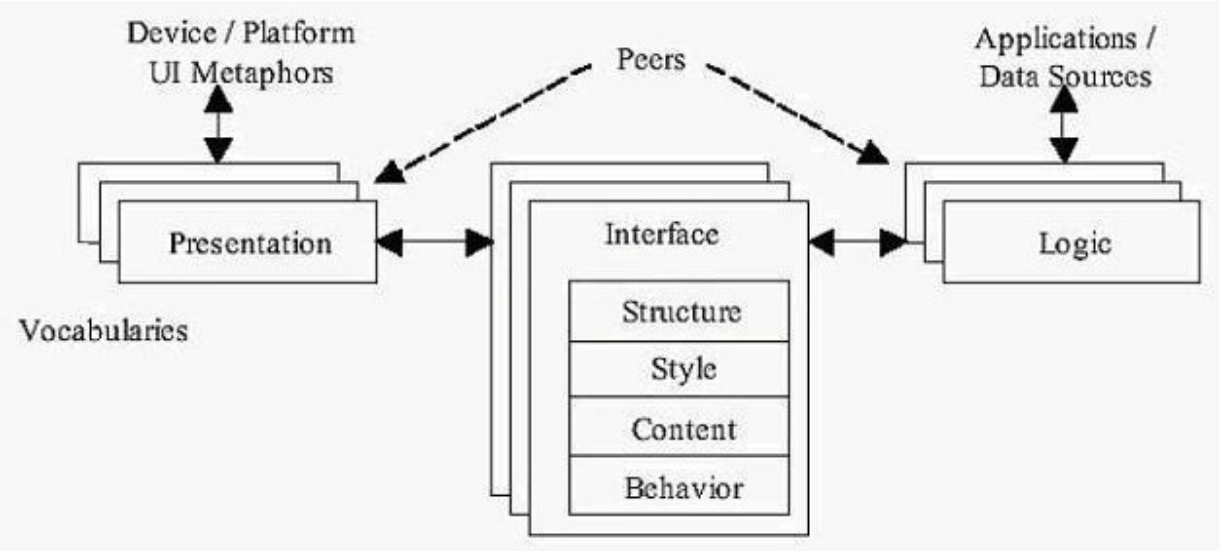

Abbildung 9: Struktur von UIML [33]

Die Struktur spiegelt sich natürlich in der UIML-Datei wider, wie nachstehender Auszug zeigt [33]:

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
 "http://uiml.org/dtds/UIML3_0a.dtd">
<uiml xmlns='http://uiml.org/dtds/UIML3_0a.dtd'>
  <head> ... </head>
  <template> ... </template>
  <interface> ... </interface>
  <peers> ... </peers>
</uiml>
```

Ein einfaches „Hello World“ Beispiel mit Mapping für WML und VoiceXML ist nachfolgend zu betrachten [33]:

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 3.0 Draft//EN"
 "http://uiml.org/dtds/UIML3_0a.dtd">
<uiml>
  <interface>
    <structure>
      <part id="TopHello">
```

```
        <part id="hello" class="helloC"/>
      </part>
    </structure>
    <style>
      <property part-name="TopHello" name="rendering">Container</property>
      <property part-name="TopHello" name="content">Hello</property>
      <property part-class="helloC" name="rendering">String</property>
      <property part-name="hello" name="content">Hello World!</property>
    </style>
  </interface>
  <peers>
    <!-- Mapping für WML -->
    <presentation id="WML">
      <d-class id="Container" maps-to="wml:card">
        <d-property id="content" maps-to="wml:card.title"/>
      </d-class>
      <d-class id="String" maps-to="wml:p">
        <d-property id="content" maps-to="PCDATA"/>
      </d-class>
    </presentation>
    <!-- Mapping für VoiceXML -->
    <presentation id="VoiceXML">
      <d-class id="Container" maps-to="vxml:form"/>
      <d-class id="String" maps-to="vxml:block">
        <d-property id="content" maps-to="PCDATA"/>
      </d-class>
    </presentation>
  </peers>
</uiml>
```

Würde nun von einem Handy auf eben genanntes Beispiel zugegriffen werden, so würde das WML Mapping verwendet werden und würde letztendlich folgende WML-Datei generieren [33]:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.0//EN"
 "http://www.wapforum.org/dtds/wml.xml">
<wml>
```

```
  <card title="Hello">
    <p>Hello World!</p>
  </card>
</wml>
```

Wie das obige Beispiel zeigt, ist für jede Zielsprache ein eigenes Vokabular notwendig. Das kann bei mehreren unterstützten Plattformen ziemlichen Aufwand bedeuten.

2.4.3 XIML

Folgende Anforderungen waren für die Entwicklung von XIML (Extensible Interface Markup Language) [38] wichtig abzudecken:

- Unterstützung von Design, Ablauf, Organisation und Evaluierungsfunktionen
- konkrete und abstrakte Datenelemente eines Interfaces nachvollziehen können
- wissensbasierten Systemen die Nutzung der Datenerfassung ermöglichen.

XIML war aus der Motivation heraus entstanden, dass es keine einheitlichen Standards für die Darstellung und Manipulation von „interaction data“ gibt. Unter „interaction data“ verstehen die Autoren von XIML Daten, welche alle relevanten Elemente eines User-Interfaces definieren und damit zusammenhängen. Der Grund warum die so genannten „interaction data“ bisher nicht effektiv umgesetzt werden konnten, liegt in der hohen Komplexität. Die Daten behandeln nicht nur konkrete Elemente wie „Widgets“, sondern auch abstrakte Elemente wie den Kontext in welchem die Interaktion stattfindet.

XIML zeichnet sich von zwei wesentlichen Punkten ab. Der erste ist die Studie der Ontologie und ihrer Darstellung und der andere ist die Arbeit von Interface Modellen. Vom ersten übernimmt XIML die Repräsentationsprinzipien und vom letzten leitet es die Typen und Eigenschaften der „interaction data“ ab. Die Basisstruktur mit ihren repräsentativen Units ist in Abbildung 10 dargestellt und wird folgend mit ihren Hauptkomponenten beschrieben. [37]

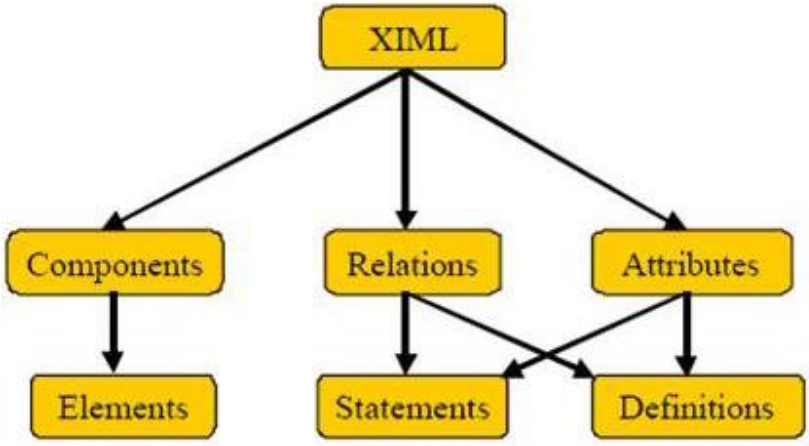

Abbildung 10: Basisstruktur von XIML [37]

- Components

  In der ersten Version von XIML gibt es fünf vordefinierte Basis-Interface-Komponenten: Task, Domain, User, Dialog und Präsentation. Die Task-Komponente übernimmt den Business-Prozess und/oder die Benutzeraufgaben, welche das Interface unterstützt. Beispiele hierfür sind: „Enter Date“, „View Map“ oder „Perform Contract Analysis“. Die Domain-Komponente ist eine organisierte Erfassung von Datenobjekten und Klassen von Objekten, welche in einer Hierarchie gegliedert sind. Diese Hierarchie ist ähnlich jener einer Ontologie. Beispiele dieser Komponente wären „Date“, „Map“ oder „Contract“. Die User Komponente definiert eine Hierarchie – einen Baum (Tree) – von Benutzern. Ein Element dieser Komponente wäre zum Beispiel „Doctor John Smith“. Bei der Präsentation wird eine Hierarchie von Interaktions-Elementen definiert, welche aus konkreten Objekten bestehen, die mit Benutzern kommunizieren. Das wäre zum Beispiel ein „Window“, ein „Button“ oder ein komplexes „Widget“. Zum Schluss fehlt noch die Dialog-Komponente, welche Elemente definiert, die Interaktions-Aktionen mit Benutzern möglich machen. Zum Beispiel „Click“ oder eine „Voice response“.

- Relations

  Eine Relation in XIML ist eine Definition oder eine Anweisung, welche zwei oder mehrere XIML Elemente entweder innerhalb einer oder über mehrere Komponenten verbindet. Zum Beispiel: „Datentyp A *wird angezeigt mit* Präsentations-Element B oder Präsentations-Element C“ (Relation ist hier kursiv dargestellt) ist eine Verbindung zwischen einem Domain-Komponenten-Element und einem Präsentations-Komponenten-Element.

- Attributes

  Attribute sind in XIML Features oder Eigenschaften von Elementen, welche einem „value“ zugewiesen werden können. Der „value“ eines Attributes kann entweder ein normaler Datentyp sein oder aber eine Instanz eines anderen existierenden Elements.

Eine der wichtigsten Anwendungsmöglichkeiten von XIML liegt in der Entwicklung von User-Interfaces, welche auf verschiedenen Geräten darstellbar sind. Das ist deswegen möglich, weil XIML eine strikte Trennung zwischen der Definition des User-Interfaces und dem Rendering eines Interfaces macht. Abbildung 11 zeigt dies anhand eines Interfaces mit zwei Endgeräten.

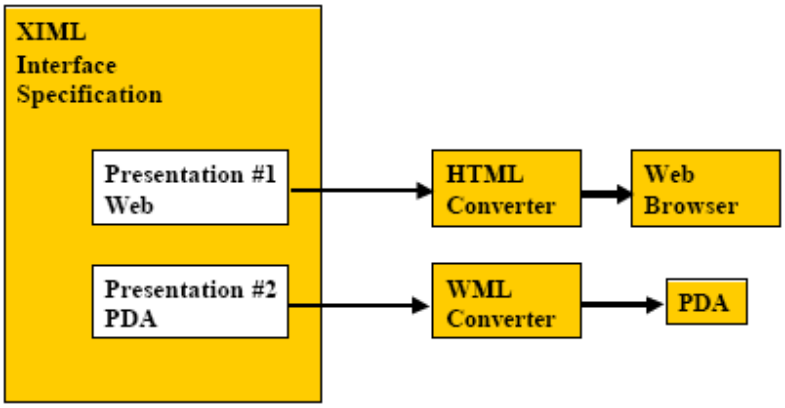

Abbildung 11: XIML Framework mit zwei Endgeräten [37]

XIML ist somit für verschiedenste Plattformen mit entsprechenden Konvertern anwendbar. Anschließend noch ein Auszug eines Beispiels [39]:

```
<?xml version="1.0" ?>
<INTERFACE ID="Ex1">
  <DEFINITIONS>
    <RELATION_DEFINITION NAME="Task_USES_Dom">
    ...
    <RELATION_DEFINITION NAME="Dial_IS_MAPPED_ONTO_PRES">
    ...
  </DEFINITIONS>
  <MODEL_COMPONENTS>
    <TASK_MODEL ID="Dictionary">
    ...
    <DOMAIN_MODEL ID="DictObjects">
    ...
    <PRESENTATION_MODEL ID="DictBasicPres">
    ...
    <DIALOG_MODEL ID="DictDialog">
    ...
  </MODEL_COMPONENTS>
</INTERFACE>
```

2.4.4 XAML

Die Markup-Sprache XAML (Extensible Application Markup Language) ist eine neu entwickelte Auszeichnungssprache, welche mit Microsoft Windows Vista [34] entstanden ist. XAML soll

ähnlich wie bei Mozilla mit XUL für das Layout Verwendung finden. Vorstellen kann man sich XAML als eine Kombination von HTML, SVG (Scalable Vector Graphics) und CSS. So wird es nicht weiter überraschen, dass die Dokumente in HTML-ähnlicher Syntax zu schreiben sind. In XAML wird gänzlich auf das von W3C standardisierte CSS verzichtet, stattdessen werden eigene Elemente verwendet. Auch die Vektor-Darstellung die XAML benutzen kann lehnt sich nicht an den Standard SVG, sondern an WVG (Windows Vector Graphics) an. Das Event-Handling und die Realisierung der Applikationslogik wird mit einer .NET mitgelieferten Programmiersprache realisiert (zum Beispiel mit C# oder VB.NET). [29]

Verwendung findet XAML bei der Programmierung des Windows Presentation Foundation Object Models, aber auch bei der Entwicklung eigener User-Interface-Applikationen. Jedes XAML-Element entspricht einer Object Model Klasse. Eventuell vorhandene Attribute dieses Elementes entsprechen wiederum den Eigenschaften der eben genannten Klasse. [35]

Folgende Vorteile sollen sich bei der Verwendung von XAML bei der Programmierung ergeben [29]:

- Bessere Sichtbarkeit von der Hierarchie der Steuerelemente
- Bessere Sichtbarkeit von der Vererbung der Attribute
- Einfache Verarbeitung und Interpretation der Auszeichnungssprache durch Tools
- Trennung von UI und Logik.

Eine XAML-Datei kann direkt wie eine Webseite über den Internet Explorer aufgerufen werden, sofern kein Code für die Realisierung der Applikationslogik oder das Event-Handling in der Datei eingebettet ist (Inline Code), oder separat in einer eigenen Datei (Code-behind). Ist aber ein solcher Code vorhanden, muss die XAML-Datei kompiliert werden. Ein Beispiel für eine XAML Datei mit dem Ergebnis ist nun folgend angegeben:

```
<Canvas
 xmlns=xmlns=http://schemas.microsoft.com/2003/xaml
 Background="LightCyan" Width="100%" Height="100%">
  <Image Source="lh.bmp" Canvas.Left="5" Canvas.Top="5"/>
  <Text Canvas.Left="90" Canvas.Top="20" FontSize="36">
    Hello, Longhorn!
  </Text>
</Canvas>
```

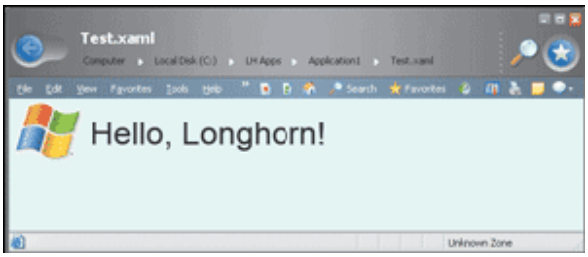

Abbildung 12: Laden des oben angegebenen Codes im Internet Explorer [29]

Das eben angeführte Beispiel enthält keinen Code für die Applikationslogik oder das Event-Handling, ist daher direkt im Internet Explorer aufrufbar (Abbildung 12).

Wäre ein solcher Code vorhanden, muss wie schon vorher beschrieben, die Datei vorher kompiliert werden. Nach dem Kompilieren wird für jede XAML-Datei eine .NET Quellcode-Datei generiert. Bei diesem Vorgang wird die XAML-Datei in der jeweiligen .NET Sprache (zum Beispiel C# oder VB.NET) abgebildet. Die XAML-Datei wird anschließend nicht mehr verwendet.

XAML beinhaltet viele Elemente zur Darstellung des UI-Layouts. Darüber hinaus bietet es auch Elemente für Animationen, Text-Formate usw. an. Es soll aber auch Online-Dokumente besser anzeigen und lesen können, wobei es sich die Vorteile von PDF und Flash ebenfalls zu Nutze macht. [29]

2.4.5 XFORMS

HTML war seinerzeit der durchschlagende Erfolg für Webanwendungen. Eingesetzt für diverse Webshops mit seinen Forms war es nicht wegzudenken. Doch nun wo andere Technologien ebenso stark vordringen, reicht HTML nicht mehr aus. Man denke zum Beispiel an eine Fahrplanauskunft via Handy. Eine Webseite, die für einen PC als auch ein Handy gut interpretiert werden kann, ist mit HTML nicht möglich. Das war einer der Gründe, ein anderer ebenso entscheidender war das Problem der Validierung. Client-seitig war dies mit HTML nicht möglich. Anfangs wirkte man mit CGI-Skripts server-seitig vor der eigentlichen Applikation entgegen. Diese Methode führte aber wieder dazu, dass bei vielen Zugriffen der Server ziemlichen Belastungen ausgesetzt war. Als nächsten Ansatz verwendete man unter anderem JavaScript um Validierungen client-seitig durchführen zu können. Auch hier konnte man nicht ganz auf eine server-seitige Validierung verzichten. Außerdem werden diverse Skriptsprachen nur teilweise oder gar nicht von einigen Browser unterstützt. [40]

Mit diesen angeführten Problemen wurde eine XForms Working Group ins Leben gerufen, die sich mit einer Neuentwicklung von HTML – XForms beschäftigen sollte. Sie stellten sich einige Anforderungen zusammen, die mit XForms umzusetzen sind. Hier ein Auszug einiger der damaligen Anforderungen [41]:

- XForms soll XML verwenden
- Forms sollen einfach an mehrere Anwender und Orte zu senden sein
- XForms soll Formdaten, Präsentation und den Zweck der Anwendung voneinander trennen
- Validierungen und Berechnungen sollen in der Sprache inkludiert sein

- XForms soll so entwickelt werden, dass Anwendungen in HTML auch in der neuen Sprache wieder umgesetzt werden können.

Mittlerweile ist bereits die XForms Spezifikation [43] mit der Version 1.0 (Second Edition) eine W3C-Empfehlung. In XHTML 2.0 [42] soll XForms standardmäßig integriert sein, bei den vorherigen Versionen muss ein eigener Namensraum für XForms definiert werden.

XForms hat folgende Eigenschaften aufzuweisen [29]:

- Trennung der Daten und der Logik von der Präsentation

  XForms hat kein eigenständiges Format, sondern muss in einem existierenden Dokument eingebettet werden, wie zum Beispiel in XHTML. Somit ist eine Trennung von der Präsentation möglich. Das heißt, dass der Container, welcher das Dokument einbettet, verantwortlich für die Präsentation ist und nicht XForms.

- Integration von XML

  XForms basiert, wie schon bei den Anforderungen erwähnt, auf XML. Die Initialisierung von Formularen kann durch lokale oder externe XML-Dateien erfolgen. Die Daten, welche im Formular eingegeben, bzw. schon standardmäßig gesetzt wurden, können nach erfolgreicher Durchführung einer Submit-Operation als XML Daten behandelt werden, aber auch wie bei HTML als Name/Werte Paare. Außerdem kann man ein Formular an mehrere Server versenden.

- Vollständigkeit der Datentypenüberprüfung

  XForms bietet weiters eine Datentypenüberprüfung an. Somit können Eingabefelder auf ihren korrekten Typ überprüft werden und bei Falscheingabe entsprechende Fehlermeldungen angezeigt werden. Durch Verwendung von CSS können falsche Eingabedaten zusätzlich visuell gekennzeichnet werden. Diese Validierungen erfolgen ohne Skripts und werden lokal ausgeführt, man erspart sich somit den Transfer zum Server und wieder zurück um zu überprüfen, ob die Eingaben korrekt waren oder nicht. XForms Prozessoren müssen eine Reihe von Datentypen unterstützen (Built-in Types). Man kann aber mit einer Integration eines XML-Schemas eigenständige Typen deklarieren und so beliebige Erweiterungen inkludieren.

- Weniger Verwendung von Skripten

  Durch die Verwendung von XPath und XML-Schema wird der Gebrauch von Skripten minimiert, bzw. macht sie überflüssig. Zusätzlich setzt XForms auf XML-Events [45] auf, die etliche Events umfassen.

In der Struktur weist XForms ebenfalls wesentliche Änderungen gegenüber HTML auf. XForms besteht aus einem XForms-Model, welches den Inhalt des Formulars beinhaltet. Ein Model besteht aus einem „model“-Element, einem „instance“-Element, dem „bind“-Element und aus einem „submission“-Element.

Ein Dokument kann mehrere dieser „model“-Elemente besitzen. Das „instance“-Element beinhaltet die Daten des Formulars. Hier werden sie nicht wie bei HTML als Name/Werte Paare, sondern als XML Daten gespeichert. Das „bind“-Element dient dazu, um Daten innerhalb einer Instanz anzusprechen (Binding). Mit diesen Elementen kann man die Daten an gewisse Datentypen binden. Dieses Element ist aber optional, da Instanzdaten ebenso über XPath referenziert werden können. Das Binding sollte aber bevorzugt verwendet werden, da bei größeren Aktualisierungen des gesamten Dokumentes lediglich die Attribute des „bind“-Elementes geändert werden müssen. Zu guter letzt kommt noch das „submission“-Element, das, wie der Name schon sagt, dass „Einreichen“ bzw. das Abschicken der gesammelten Formulardaten veranlasst. [41]

Nun zur Benutzeroberfläche, welche plattformunabhängige Darstellungen zulässt, da alle Elemente in XForms abstrakt definiert sind. Man wird also keine speziellen Elemente wie eine „List Box“ finden, sondern nur Elemente wie „select“, welche noch keine konkrete Aussage über die Darstellung machen. Hierfür gibt es spezielle Attribute, die zum Beispiel über CSS gesetzt werden können. [40]

Zu erwähnen sei noch, dass XForms client- und server-seitig einsetzbar ist. Es kann so zum Beispiel als XHTML-Datei gespeichert und von einem Browser angezeigt werden, oder über einen Server abrufbar sein. Es gibt zurzeit einige Projekte, die sich mit dem server-seitigen Einsatz von XForms beschäftigen. Unter anderem sei hier das Chiba-Projekt [48] oder das Projekt von Orbeon [49] zu erwähnen.

Abschließend noch ein Beispiel, bei dem zwei Instanzdaten („ref“ und „bind“) über ein Binding und eine XPath-Referenz von jeweils einem „output“-Element angesprochen werden.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xforms="http://www.w3.org/2002/xforms">
  <head>
    <xforms:model>
      <xforms:instance xmlns="">
        <data>
          <ref>Reference example</ref>
          <bind>Binding example</bind>
        </data>
      </xforms:instance>
      <xforms:bind nodeset="/data/bind" id="bind"/>
```

```
      <xforms:submission action="..." method="post" id="submit"/>
    </xforms:model>
  </head>
  <body>
    <xforms:group>
      <xforms:output ref="/data/ref">
        <xforms:label>Output: </xforms:label>
      </xforms:output>
      <br/>
      <xforms:output bind="bind">
        <xforms:label>Output: </xforms:label>
      </xforms:output>
      <br/>
      <xforms:submit submission="submit">
        <xforms:label>Submit</xforms:label>
      </xforms:submit>
    </xforms:group>
  </body>
</html>
```

In einem Browser mit geeignetem XForms Plug-in sieht die Ausgabe, wie in Abbildung 13 ersichtlich, folgendermaßen aus:

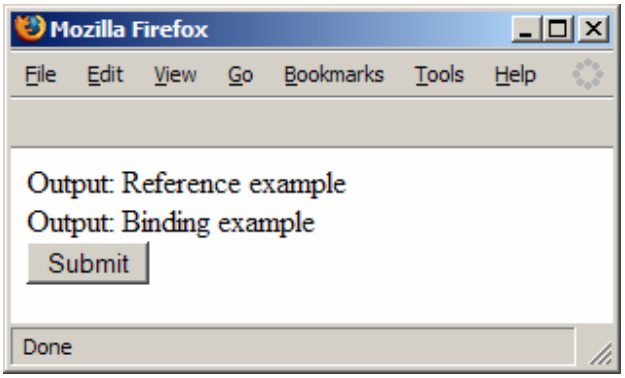

Abbildung 13: Ausgabe des obigen demonstrierten XForms Beispiels in Mozilla Firefox

2.4.6 Andere Markup-Sprachen

Es gibt eine Vielzahl an Sprachen die annähernd den gesuchten Eigenschaften entsprechen würden. Müsste man diese nun alle genauer analysieren, würde das den Umfang der Arbeit sprengen. Somit beschränkt man sich hier auf die Aufzählung zusätzlicher Markup-Sprachen mit geräteunabhängigem User Interface Design [11]:

- Alternate Abstract Interface Markup Langugae (AAIML)
- Abstract User Interface Markup Language (AUIML)
- SAP Web Dynpro Protocol
- SEESCOA Project
- User Interface extensible Markup Language
- W3C Device Independence Working Group (DIWG)

3 Human Resource Management

Dieses Kapitel beschreibt, warum HR für SemCrypt angedacht wurde. Weiters wird auf die Verwendung von HR-XML, welches ein standardisiertes Vokabular für das Personalwesen darstellt und sich daher ideal eignet, eingegangen. Außerdem werden noch Szenarien für den HR-Bereich aufgezeigt, die in SemCrypt Verwendung finden sollen.

3.1 Motivation

Wie bereits in Kapitel 2.3.2 gezeigt wurde, soll SemCrypt für den Bereich Humanressourcen verwendet werden. Dieser Bereich ist für Unternehmen von enormer Wichtigkeit. In der modernen Auffassung der Ökonomie gilt der Mensch als wertvollstes „Gut“ im Betrieb. Da aber der Mensch, ebenso wie „andere“ Güter, Kosten verursacht (in diesem Fall Lohnkosten), ist es natürlich ein Ziel eines Unternehmens diese zu verringern. Möglichkeiten wären unter anderem gewisse Arbeitsbereiche zu automatisieren oder auch Reorganisationen. „Die richtige Person zur richtigen Zeit am richtigen Arbeitsplatz“ ist auch heute noch der Leitsatz für das Personal- oder Human Resource Management. Der Bedarf an geeigneten Arbeitskräften ist ein ständiges Wechselspiel und unterliegt fortwährend Angebot und Nachfrage. Diese Dynamik erfordert daher auch eine laufende Weiterentwicklung der Organisation, wie z. B. der Investition in die Weiterbildung des Personals. [66]

Um nun z. B. Mitarbeiter gezielt einsetzen zu können, bedarf es eben dieser Weiterentwicklung der Organisation, welche auch die Verfügbarkeit und Aktualisierung der Personaldaten ihrer Mitarbeiter beinhaltet. Diese Daten enthalten nicht nur die Namen und Adressen jener Personen, sondern auch deren Kompetenzen, die bisherigen Tätigkeiten im Unternehmen und viele weitere Informationen, unter anderem auch der Skills, welche eben für ein gezieltes Einsetzen notwendig sind.

Hier kommt HR-XML zum Einsatz, wobei diese Daten vom HR-XML-Konsortium spezifiziert werden. Das Konsortium ist eine unabhängige, keinen Gewinn anstrebende Vereinigung, die den unternehmensweiten Datenaustausch im Personalwesen standardisieren möchte. Hauptaugenmerk liegt in der Entwicklung eines standardisierten XML-Vokabulars im Personalwesenbereich. Derzeit wird das Konsortium durch Mitglieder in 22 Ländern repräsentiert. [13]

Einige fertige HR-XML Spezifikationen sind unter anderem die „Contact Method“, welche Informationen wie Adressen für Email und Homepage aber auch Telefonnummern beinhaltet oder die „Competencies“, welche die Kompetenzen von Personen, wie zum Beispiel die Führerscheingruppen oder die Programmierfähigkeiten (siehe nachfolgenden Auszug eines HR-XML-Dokuments), inne hat.

```
...
<ContactMethod>
  <Mobile smsEnabled="true">
    <InternationalCountryCode>43</InternationalCountryCode>
```

```
    <SubscriberNumber>1234567</SubscriberNumber>
  </Mobile>
  <InternetEmailAddress>mustermann@email.at</InternetEmailAddress>
  <InternetWebAddress>http://www.mustermann.at</InternetWebAddress>
</ContactMethod>
...
<Competencies>
  <Competency name="Fuehrerschein A,B">
    <CompetencyEvidence dateOfIncident="1993-07-27" name="Pruefung">
      <EvidenceId id="1932226" idOwner="Bezirkshauptmannschaft Wien"/>
      <StringValue>bestanden</StringValue>
    </CompetencyEvidence>
  </Competency>
</Competencies>
...
```

3.2 HR-Szenario

SemCrypt wird hauptsächlich für Abfragen (XPath) aus dem HR-Bereich Verwendung finden. Das Erstellen, Modifizieren aber auch das Entfernen eines Datensatzes (XUpdate:delete, XUpdate:insert, XUpdate:append, XUpdate:update) wird eher die Ausnahme als die Regel sein, da Rekrutierungen neuer Mitarbeiter, Entlassungen usw. nicht an der Tagesordnung stehen, sondern eher selten auftreten werden. Die Zugriffe auf den Inhalt der Daten sind außerdem vom Berechtigungsgrad abhängig. So wird ist es nur wenigen Auserwählten in einer Firma möglich sein, Daten abzufragen bzw. auch zu ändern. Abfragen könnten z. B. sein: Welche Mitarbeiter haben die Kompetenz (Competency) Java-Programmieren; An welchen Mitarbeiter wurde das Firmenhandy mit der Nummer (SubscriberNumber) xy ausgegeben; usw. Mit Hilfe der XUpdate-Funktionen kann man diese Daten um weitere einfach erweitern, modifizieren oder sogar löschen. Generell sieht die XPath- und XUpdate-Anweisung folgendermaßen aus:

XPath:

- <selector> (“[” (<element> “[” <attribute> “ ” <operator> “ ” <value> (“ ” “and“ “ ” <attribute> “ ” <operator> “ ” <value>)* “]”)* (“ ”? “and“? “ ”? “.” “[” “.” “ ” <operator> “ ” <value> (“ ” “and” “ ” <element> “ ” <operator> “ ” <value>)* “]”)? “]”)?

  Das “Selector”-Element wird in Kapitel 5.2.3 - XForms:Model für Submit – näher vorgestellt. Alle Element/Attribut-Operator-Value Felder (z. B. /xpath[./@attribute eq “value“]) haben einen XPath „<element>“ vorangestellt. Eine Ausnahme bildet das Element, welches mit dem „Selector“ ident ist. Dieses benötigt nur einen „.“ für den aktuellen Pfad (z. B. .[. eq „value“]).

XUpdate:

- “<xupdate:modifications version="1.0" xmlns:xupdate="http://www.xmldb.org/xupdate">” (<xupdate select> (<attribute/element name> <value> </attribute/element>) | (<value>) </xupdate>)+ </xupdate:modifications>”

  Das “<xupdate select>” Tag gibt die Art des XUpdates an (insert, delete, …) inklusive des Select-Attributes, mit welchem ausgewählt wird, wo eingefügt, gelöscht, etc. werden soll. Dann wird je nach XUpdate-Anweisung ein Value oder bei Insert und Append das neue Attribut/Element mit Namen und Value angegeben.

Die zuvor angesprochene Berechtigungsvergabe wird durch XAccess definiert. So werden vorerst verschiedene Rollen (Projektleiter, Gruppenleiter, …) in einem Unternehmen definiert und danach die Mitarbeiter diesen Rollen zugeteilt. Es kann daher auch sein, dass ein Mitarbeiter mehreren Rollen zugeteilt ist. Jeder Rolle können dann je Zugriffspfad verschiedene Berechtigungen zugeteilt werden. Im Folgenden wird ein Beispiel einer Berechtigungsvergabe auf ein Element und ein weiteres auf ein Attribut für die Rollen Gruppenleiter (GL) und Projektleiter (PL) gezeigt:

```
...
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo/PersonName</PathValue>
  <RoleName>GL</RoleName>
  <RoleName>PL</RoleName>
  <RightName>read</RightName>
  <RightName>delete</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                              /ContactMethod/Mobile/@smsEnabled</PathValue>
  <RoleName>GL</RoleName>
  <RightName>read</RightName>
</Permission>
```

Im ersten Fall haben beide Rollen Lese- und Löschrechte auf das Element PersonName. Im zweiten Fall darf nur der Gruppenleiter das Attribut „smsEnabled“ lesen. So wie in diesem Beispiel werden ebenso alle anderen möglichen Pfade (HR-Komponenten) definiert und mit Berechtigungen versehen, die alle dem XAccess-Schema zu Grunde liegen.

In SemCrypt gebräuchliche HR-Komponenten sind unter anderem:

- PersonInfo

  Diese Komponente setzt sich aus PersonId, PersonName, ContactMethod und PersonDescriptors zusammen. Der PersonName gliedert sich in LegalName, GivenName, PreferredGivenName, MiddleName, FamilyName und Affix, also in jene Daten, die alle möglichen Namen einer Person bestimmen inklusive ihres Affix'. Zusätzlich mit der PersonId ist somit eine Person eindeutig bestimmt. Die ContactMethod beinhaltet die Kontaktdaten der jeweiligen Person wie Adresse, Telefonnummer, Fax, Emailadressen, usw. Der PersonDescriptor enthält vorwiegend biologische und demographische Daten. Dies sind z. B. Haar- und Augenfarbe, Gewicht, Alter, Nationalität, Geschlecht, etc. Mit PersonInfo lassen sich somit alle personenspezifischen Daten festhalten. In wie weit diese Daten in einer Firma notwendig sind, sei dahingestellt.

- Competencies

  Die Kompetenzen sind im HR-Bereich von essentieller Bedeutung. Diese Daten beinhalten Ausbildungen, Schulungen, Führerscheine und vieles mehr, eben alle Kompetenzen einer Person. Diese Angaben können mit dem jeweiligen Datum der erbrachten Leistung versehen werden, außerdem sind Gewichtungen (Anfänger, Profi, …) ebenfalls möglich. Anhand dieser Daten ist es einem Unternehmen möglich ihre Mitarbeiter gezielter auszubilden und einzusetzen.

- JobPositionHistory

  Wie der Name schon sagt, sind in dieser Komponente alle bisherigen Jobs mit Zeitangaben einer Person enthalten. Auch verschiedene Positionen die man in einer Firma inne hatte werden hier eingetragen. Dies kann man mit einem Lebenslauf vergleichen. Diese Daten spiegeln die Erfahrungswerte eines Mitarbeiters wider und können außerdem zum Eruieren des Einstiegsgehalts dienen.

- Salary

  Lohn- und Gehaltsabrechnungen sind Gegenstand jedes Unternehmens. Salary besteht aus Daten wie Kontonummer, Betragswerten und anderen Eigenschaften eines Mitarbeiters.

- AssessmentResults

  Mit dieser Komponente lassen sich Leistungsbewertungen festhalten. Abgeschlossene als auch nicht abgeschlossene Tätigkeiten sind hier möglich. Den Fortschritt einer Tätigkeit kann man über ein Status-Feld anpassen.

Wie man aus den vorher gezeigten Komponenten gesehen hat, ist deren Informationsgehalt ziemlich weitläufig. Dies lässt natürlich weit reichende Verwendung zu. Da aber SemCrypt auch für andere Bereiche angedacht ist, kommt dies dem Projekt sehr entgegen. Biologische Daten wie das Gewicht haben in einem Unternehmen eigentlich nichts zu suchen, hingegen bei einem Einsatz von eHealth sehr wohl.

4 Entwurf

In diesem Kapitel werden die Ansätze und Entscheidungen für den im späteren Kapitel beschriebenen Lösungsweg aufgezeigt.

4.1 Vorgaben

In der Einleitung wurde das Ziel der Arbeit bereits erläutert. Hier folgt nun eine detailliertere Beschreibung der Ziele und ihrer Vorgaben. Das Ziel, nochmals zusammenfassend, besteht darin in ein bestehendes Projekt namens SemCrypt einen Generator einzuarbeiten, dessen Aufgabe es sein soll, ein geräteunabhängiges Web-Interface zu generieren. Das Web-Interface soll einem Benutzer Datenbankabfragen auf verschlüsselte XML-Daten ermöglichen. Die Entschlüsselung ist Aufgabe des Encryption-Plug-ins (siehe Kapitel 5.1). Die Struktur dieser XML-Daten entspricht einem angepassten HR-XML Standard, das heißt, dass bestehende HR-XML-Schemadaten um die für das Projekt relevanten Elemente erweitert wurden.

Dem Generator steht eine XML-Datei (siehe nachfolgendes beispielhaftes XAccess-XML) mit zugehörigem Schema, welches in Kapitel 2.1.6 bereits vorgestellt wurde, zur Verfügung. Diese XML-Datei wird in weiterer Folge XAccess-Datei genannt. Es beinhaltet nämlich die Zugriffsdaten für die einzelnen Benutzerrollen, die im System vorkommen können (z. B. Unternehmensführung, Projektleiter, Gruppenleiter, etc.), die entsprechenden XPaths, welche den absoluten Pfad in der Datenbank zu dem jeweiligen Element bzw. Attribut definieren, sowie die Security-Pfade mit den jeweiligen Restriktionen für einen XPath.

```
<?xml version="1.0" encoding="UTF-8"?>
<XAccess xmlns="http://semcrypt.ec3.at/xml/access/types"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://semcrypt.ec3.at/xml/access/types
         xaccess.xsd">
<!-- SECURITY ROLES -->
  <SecurityRoles>
    <SecurityRole>
      <Description>Unternehmensfuehrung</Description>
      <RoleName>UF</RoleName>
    </SecurityRole>
    ...
  </SecurityRoles>
  <!-- SECURITY PATHS -->
  <SecurityPaths>
    <SecurityPath>
      <Description>Element Competency</Description>
```

```
      <PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
      <Type>anonymous</Type>
    </SecurityPath>
    <SecurityPath>
      <Description>Attribut smsEnabled</Description>
      <PathValue>/SemCryptEmployeeInfo/PersonInfo
                            /ContactMethod/Mobile/@smsEnabled</PathValue>
      <Type>boolean</Type>
    </SecurityPath>
    ...
  </SecurityPaths>
  <!-- SECURITY RIGHTS -->
  <SecurityRights>
    <SecurityRight>
      <Description>read operation allowed</Description>
      <RightName>read</RightName>
    </SecurityRight>
    ...
  </SecurityRights>
  <!-- PERMISSIONS -->
  <Permissions>
    <Permission>
      <PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
      <RoleName>UF</RoleName>
      <RightName>read</RightName>
      <Restrictions>
        <Path>
          <Operation>eq</Operation>
          <Value>type</Value>
        </Path>
      </Restrictions>
    </Permission>
    ...
  </Permissions>
</XAccess>
```

Der Generator soll mit dieser Information ein Admin-GUI erstellen, welches alle Securitypaths (XPaths) je SecurityRole (Rolle) mit entsprechenden Permissions beinhaltet. Es werden also nur jene XPaths (Elemente bzw. Attribute) im Admin-GUI aufgelistet, die auch unter Permission einen Eintrag besitzen. Das erstellte Admin-GUI dient dann zum Auswählen der möglichen Datenbankabfragen (XPaths) und zum Erstellen zusätzlicher Restriktionen für eine Datenbankabfrage (XPath/XUpdate). Zusätzlich deswegen, weil alle bereits bestehenden Einschränkungen in der XAccess-Datei übernommen wurden. Einzelne Datenbankzugriffsmöglichkeiten, die im Admin-GUI gesetzt werden, können auch wieder entfernt werden. Gleiches gilt auch für die Restriktionen. Außer für XPath-Restriktionen, die bereits in der XAccess-Datei getroffen wurden. Diese können nicht wieder entfernt werden – sie sind verpflichtend. Das heißt, wählt nun ein Administrator einen XPath aus um diesen als möglichen Datenbankzugriffspfad zu erlauben, so gelten auf jeden Fall die Einschränkungen aus der XAccess-Datei. Er kann aber zusätzliche Einschränkungen für den gewählten XPath vornehmen, muss aber nicht. Wäre zum Beispiel ein XPath vom Typ „double“, könnte dieser bereits eine vordefinierte Einschränkung in der XAccess-Datei besitzen, dass alle Werte größer null sein müssen. Im Admin-GUI lässt sich dieser Eintrag nicht mehr löschen, man kann aber den positiven Bereich mit einem Eintrag, dass alle Werte kleiner zwanzig sein sollen verkleinern. Somit wären in diesem Beispiel nur Werte zwischen null und zwanzig gültig. XUpdate-Restriktionen (Lösch-, Update-, Insert- und Append-Rechte) können im Admin-GUI entfernt werden (sofern vorhanden), in der XAccess-Datei nicht definierte Rechte je XPath können aber nicht hinzugefügt werden. Wenn also ein XPath Lese- und Update-Rechte besitzt, kann man das Update-Recht löschen bzw. wieder hinzufügen. Man kann aber z. B. keine Löschrechte darauf vergeben. Die Leserechte sind nicht entfernbar, eine Nichtselektierung des jeweiligen XPaths bringt hierfür den gewünschten Erfolg.

Sind alle Auswahloptionen getroffen, wird letztendlich das Web-Interface (GUI) erstellt. Das generierte GUI soll für verschiedenste Geräte geeignet sein, das heißt, es muss eine Sprache gewählt werden, die eine Geräteunabhängigkeit ermöglicht und die ausgewählten XPaths grafisch wiedergibt. Der Benutzer kann nun anhand der Auswahlmöglichkeiten eine Datenbankabfrage starten (zum Beispiel: Wie viele Personen haben die Kompetenz Java-Programmieren). Das ganze bildlich zusammengefasst zeigt Abbildung 14:

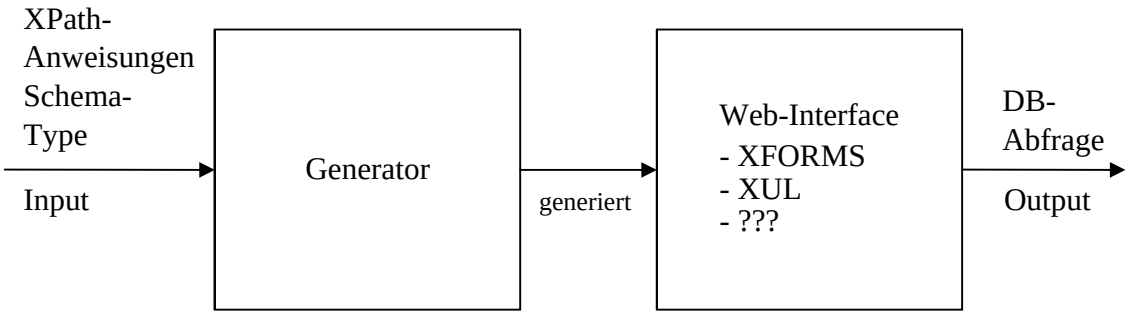

Abbildung 14: SemCrypt UI Generator

4.2 Generator

Die wichtigste Aufgabe des Generators ist die Generierung des Web-Interfaces. Wie bereits erwähnt, sollen aber auch zusätzliche Auswahlmöglichkeiten, wie z. B. das Setzen neuer Restriktionen, geschaffen werden. Da bisherige Komponenten des SemCrypt Projekts auf Java basieren, fällt die Entscheidung der Entwicklung des Generators ebenfalls auf Java. Als Entwicklungsumgebung dient hier Eclipse IDE, da sich dieses anhand der Plug-in Erweiterbarkeit ideal eignet, um den Generator als solches zu realisieren. Außerdem ist es im Projekt bereits in Verwendung. Somit ist der Generator auf jedem System, welches Eclipse erfolgreich installieren kann, lauffähig. Lediglich auf die Eclipse-Version ist zu achten.

Aufgrund einer baumförmigen XML-Struktur können die Auswahlmöglichkeiten der einzelnen XPaths grafisch als Baum (Tree) realisiert werden, da XPaths Zugriffsmöglichkeiten auf XML-Dokumente darstellen und sich somit einfacher und logischer darstellen lassen.

4.3 Web-Interface

Beschreibungssprachen für Benutzeroberflächen, so genannte UIDLs (User Interface Description Language) werden schon seit geraumer Zeit entwickelt. Ziele dieser Sprachen sind unter anderem Plattformunabhängigkeit, Wiederverwendbarkeit und Geräteunabhängigkeit, also jene Aspekte die für das Web-Interface benötigt werden. Diese Ziele sind aber extrem schwierig zu erreichen. Man denke dabei an die verschiedenen Plattformen mit ihren unterschiedlichen Systemressourcen und Bibliotheken.

Hier kommen nun die Markup-Sprachen ins Spiel, welche auch Auszeichnungssprachen genannt werden. Sie dienen der Beschreibung von Daten und Verfahren, welche zur Bearbeitung dieser Daten notwendig sind. Markup-Sprachen werden oft in deskriptive und prozedurale Markup-Sprachen unterteilt, die präsentationsbezogene Markup-Sprache wird ebenfalls manchmal miteinbezogen. Unter erstere Sprache fallen vor allem die XML definierten Sprachen wie HTML, WML und SVG hinein und prozedurale Sprachen sind zum Beispiel PDF und PostScript. [15] Außerdem basieren sie alle auf XML und können somit problemlos auf verschiedene Plattformen portiert werden. Mit ihnen können Benutzeroberflächen generiert werden, wobei die Struktur der Oberfläche von der Logik getrennt behandelt wird. [29]

Als Ansatz werden verschiedene geräteunabhängige Markup-Sprachen [11] die bereits im Kapitel Web-Interface Systeme dargestellt wurden, auf ihre Tauglichkeit untersucht, wobei HTML sich als idealer Vorreiter in Web-Interfaces als Markup-Sprache erwiesen hat.

4.4 Vergleich der ausgewählten Markup-Sprachen

Um eine Auswahl bezüglich der Sprache treffen zu können, werden unter diesem Punkt alle bisher besprochenen Markup-Sprachen in den wichtigsten Punkten miteinander verglichen. Außerdem werden die wichtigsten Vor- und Nachteile jeder Sprache aufgezeigt. Siehe dazu Tabelle 1.

UIML und XIML haben den wesentlichen Nachteil, dass deren letzte Aktualisierung ziemlich weit zurückliegt, obwohl es nun einen Vorstoß von OASIS gibt, welche beginnend von der UIML 3.0 Spezifikation, eine Spezifikation für abstrakte Meta-Sprachen erstellen wollen. [11]

Weiters sind Teile beider Produkte kommerziell und die Verbreitung beider Technologien sehr gering, daher wahrscheinlich auch der neue Weiterentwicklungsansatz von OASIS. Die vorhandenen Beispiele sind ebenfalls nicht aussagekräftig und Dokumentationen Mangelware.

XUL und XAML sind beide zukunftsträchtige Sprachen. XUL ist aber im Gegensatz zu XAML plattformunabhängig, jedoch nur auf Mozilla-Browsern lauffähig. Die Entwicklung von XAML kann derzeit noch nicht wirklich vorhergesehen werden, da es erst mit dem neuen Betriebssystem Windows Vista ausgeliefert wird. Die Sprache ist wie bereits erwähnt, nur auf Windows Systemen lauffähig. Trotzdem muss das für die Verbreitung dieser Sprache keine negativen Auswirkungen haben, da der Anteil an Microsoft Windows Anwendern immer noch ziemlich groß sein dürfte und somit eine sehr gute Verbreitung und einen hohen Bekanntheitsgrad mit sich bringen könnte. Mozilla hat hier sicherlich zurzeit das Nachsehen, aber auch der bekannte Mozilla Browser Firefox etabliert sich immer besser.

XForms ist bereits ziemlich gut im Internet verbreitet. Es gibt zahlreiche Beispiele, die auf verschiedensten Browsern lauffähig sind. Damit aber eine Lauffähigkeit garantiert ist, müssen auf diversen Browsern vorher Plug-ins installiert werden. Der derzeitige Entwicklungsstand der Browser Plug-ins ist ziemlich unterschiedlich. Eine komplette Implementierung der XForms 1.0 Spezifikation ist bei keinem Plug-in gegeben. Dadurch das XForms in XHTML 2.0 standardmäßig inkludiert ist, dürfte eine Verbreitung dieser Sprache ziemlich realistisch sein und Plug-ins zukünftig obsolet machen. Durch die hervorragenden Validierungsmöglichkeiten wird XForms als Nachfolger von HTML gehandelt.

Es ist nicht leicht eine Entscheidung über eine Sprache zu treffen, die auch noch über Jahrzehnte hinweg Einsatz finden sollte. In einer kurzlebigen Zeit wie dieser wahrscheinlich auch unmöglich! T. V. Raman schreibt in seinem Buch „XForms – XML Powered Web Forms“ [40], dass sich das abstrakte User-Interface-Vokabular, wie es XForms verwendet, bewähren wird. Zumindest solange wie HTML von 1993 bis jetzt überlebt hat. Wie auch immer, man muss sich trotzdem aus den ausgewählten Technologien für XForms entscheiden. Wie schon vorher erwähnt, bietet es zahlreiche Beispiele als auch Artikel im Internet an. Kostenlose Verfügbarkeit, bis auf einzelne Browser Plug-ins, sowie Standardisierung und Open Source sprechen für sich. Es setzt als Nachfolger von HTML seinen Weg fort, aber ob es sich letztendlich gegen XAML von Microsoft durchsetzen kann, sei dahingestellt.

	XUL [29][46]	UIML [29][46]	XIML [47]	XAML [29][46]	XFORMS [29]
Organisation	Mozilla	Harmonia	RedWhale	Microsoft	W3C
Standardisierung	nein	ja	nein	nein	ja
Open Source	ja	teilweise	teilweise	nein	ja
Kosten	kostenlos	kostenlos/ kommerziell	kostenlos/ kommerziell	kommerziell	kostenlos
Unterstützte Browser	Mozilla	derzeit eigener Browser notwendig	keine Angaben	Internet Explorer	derzeit Plug-ins notwendig
Sprachen/Skripte	JavaScript	theoretisch mit allen Sprachen	XIML	C#, VB.NET	JavaScript
Plattform-unabhängigkeit	ja	ja	ja	nein	ja
Dokumentation	gut	veraltet	veraltet	mittel	gut
Vor-/Nachteile	Vorteile: - Plattform-unabhängigkeit - leicht portierbar - einfache Anpassung einzelner Views (Skins) Nachteile: - nur für Plattformen mit Mozilla lauffähig - derzeit nur für Webapplikationen einsatzfähig	Vorteile: - sehr ausgereift - sehr gut erforscht - gute Werkzeug-unterstützung Nachteile: - keine abstrakteren Modelle für frühe Phasen - geringe Verbreitung	Nachteile: - allgemeiner Ansatz bringt hohen Aufwand - teilweise unterspezifiziert - keine Werkzeug-unterstützung - nicht frei verfügbar - wenig verbreitet	Vorteile: - beliebige Skalierung und Rotation von Text, Grafik und Animation Nachteile: - nur für Vista-Anwendungen - nicht für Webapplikationen einsetzbar - noch keine speziellen Toolkits einsatzfähig	Vorteile: - Trennung der Darstellung von Daten und Logik - Daten werden in XML gesammelt und können als XML-Dokument gesendet werden - Weniger Skripting notwendig - Integrierte Validierung Nachteile: - derzeit Plug-ins notwendig

Tabelle 1: Vergleich der Markup-Sprachen

4.5 Tieferer Einblick in die ausgewählte Markup-Sprache

In Kapitel 2.4.5 wurde XForms bereits vorgestellt. Hier werden nun Unterscheidungen zwischen HTML und XForms aufgeführt und die wichtigsten Formen von XForms erläutert. Die Validierung ist ebenfalls Gegenstand dieses Kapitels. Außerdem wird auf den Unterschied zwischen server- und client-seitiger Implementierung näher eingegangen, sowie der Installation beider Varianten.

4.5.1 Unterscheidung zwischen HTML und XForms

Die wichtigsten Kriterien von XForms, die die wesentlichsten Unterschiede zu HTML darstellen, wurden bereits vorgestellt. Ein weiterer Punkt ist, dass gesendete Formulardaten in XForms als XML-Daten versendet werden können. Also nicht mehr nur wie bei HTML über Schlüssel/Werte Paare. Dies kann anhand des Beispiels in Abbildung 13 gezeigt werden, indem durch Abschicken der Formulardaten und vorheriger sinnvoller Befüllung des „action“-Attributs (z. B. mit localhost:8080) mit der Methode *post* folgender HTTP-Request erstellt wird:

```
POST / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword,
application/x-shockwave-flash, */*
Accept-Language: de-at
Content-Type: application/xml; charset=UTF-8
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
nxforms/1.00; .NET CLR 1.1.4322)
Host: localhost:8080
Content-Length: 159
Connection: Keep-Alive
Cache-Control: no-cache
<data xmlns="" xmlns:xforms="http://www.w3.org/2002/xforms">
  <ref>Reference example</ref>
  <bind>Binding example</bind>
</data>
```

Die Formulardaten sind im „data“-Tag ersichtlich. Im Vergleich dazu eine herkömmliche Versendung der Daten über die Methode *get* bei welcher die Daten in Schlüssel/Werte Paare aufgegliedert sind:

```
GET /?ref=Reference%20example;bind=Binding%20example HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword,
application/x-shockwave-flash, */*
Accept-Language: de-at
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1;
nxforms/1.00; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Host: localhost:8080
Connection: Keep-Alive
```

Aber auch die Validierung ist zwischen den beiden Markup-Sprachen anders. Während sie bei HTML über den Server erfolgt (mit JavaScript auch client-seitig lösbar), wird bei XForms im Client über den XForms-Prozessor validiert (Voraussetzung: XForms entweder als Plug-in oder standardmäßig am Client installiert). Dieser bedient sich beim Prüfungsprozess der Built-in Datentypen. Selbstdefinierte Datentypen und jene, die nicht in XForms inkludiert sind, müssen mit einem XML-Schema definiert werden, auf welches der Prozessor Zugriff hat und letztendlich für die Validierung heranzieht. Diese Unterscheidungen zusammengefasst sind:

- HTML-Processing

  Input-Daten am Client werden heutzutage so weit wie möglich auch dort vorvalidiert und dann erst am Server, um diesen zu entlasten und Zeit zu sparen. Client-seitig wird oftmals JavaScript eingesetzt, was aber eine Erlaubnis der Ausführung von Skripts nach sich zieht. Da aber alle Validierungen nicht am Client durchführbar sind, müssen die verbleibenden Checks am Server durchgeführt werden. Wird hier ein Fehler erkannt, so wird die gesamte Form am Client neu geladen, was unter Umständen zeitintensiv sein kann. Bei erfolgreicher Überprüfung der Daten werden diese dann an das Ziel weitergeschickt. Handelt es sich z. B. um eine XML-Datenbank, müssen die Daten noch in ein für die Datenbank brauchbares XML-Format umgewandelt werden.

- XForms-Processing

  Hier werden die Input-Daten am Client validiert. Das ist durch die Built-in Datentypen, durch Vergabe so genannter „constraints“ und „relevants“ möglich. Auf die Validierung wird später in Kapitel 4.5.3 näher eingegangen. Somit entfällt hier die server-seitige Nachvalidierung. Nach erfolgreicher Überprüfung der Daten werden diese an den Webserver gesendet, welcher die Daten z. B. an eine XML-Datenbank weiterleitet. Gerade für diese Datenbanken ist XForms sehr gut verwendbar, da die Formdaten in XML-Format gesendet werden können und eine Umformatierung somit wegfällt.

Grafisch sind die Unterscheidungen in Abbildung 15 ersichtlich:

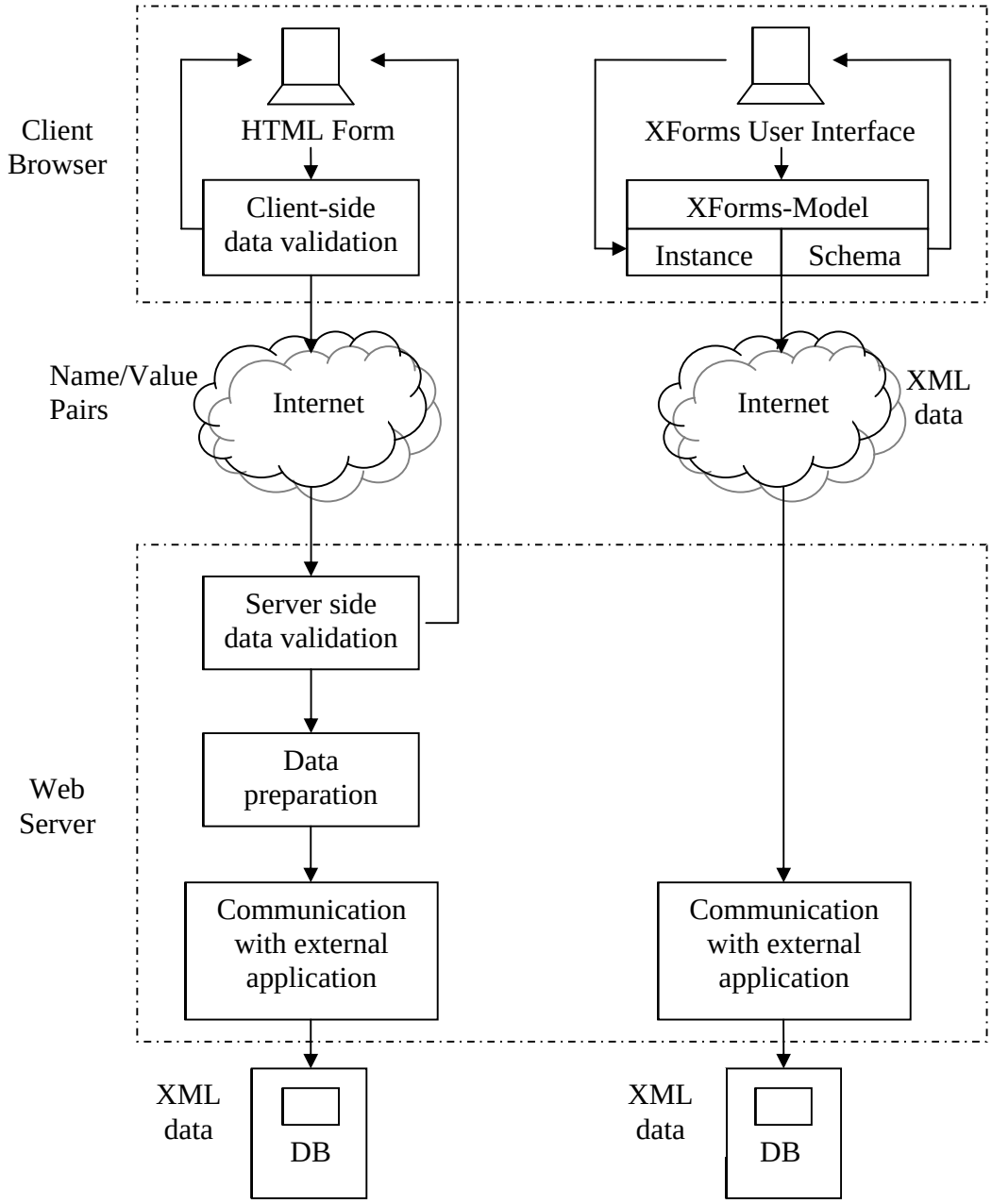

Abbildung 15: Traditionelles Form Processing vs. XForms [53]

4.5.2 XForms Formen

- input

Ähnlich wie bei HTML lässt ein Input-Feld Eingaben von Buchstaben und Ziffern zu, mit dem wesentlichen Unterschied, dass die Eingaben validiert werden können. Außerdem ist die visuelle Darstellung nicht auf eine Editierungsbox beschränkt, es kann z. B. beim Datentyp

„date“ die Box als Datumsfenster angezeigt werden. Nachfolgend ein Beispielcode mit zwei Datentypen (xsd:string und xsd:date) mit der visuellen Darstellung in Abbildung 16. [41]

```
// Referenz zu einem String Datentypen
<input ref="string">
  <label>xsd:string</label>
</input>
// Referenz zu einem Date Datentypen
<input ref="date">
  <label>Ship By:</label>
</input>
```

Abbildung 16: Darstellung einer Input-Box mit unterschiedlichen Datentypen [41]

- secret

Wie bei HTML handelt es sich hier um eine Eingabebox, in welcher die eingegebenen Buchstaben bzw. Ziffern nicht angezeigt werden. Die Daten sind aber nicht verschlüsselt und bedürfen somit weiteren Schutzmechanismen, wenn erforderlich. Siehe dazu den Beispielcode mit anschließender Darstellung in einem Browser (Abbildung 17). [41]

```
<secret ref="/session/password">
  <label>Password</label>
</secret>
```

Abbildung 17: Passwort Form [41]

- output

  Das ist die einzige Form, welche keinen Input eines Anwenders akzeptiert. Diese Form wird benutzt, um Daten eines Models als Text auszugeben, wie das folgende Beispiel zeigt. [41]

```
<output ref="/my:employee/my:name">
  <label>Name:</label>
</output>
<input ref="/my:employee/my:name">
  <label>Edit the name here:</label>
</input>
```

- trigger

  Diese Form hat Ähnlichkeiten mit HTML-Buttons. In einem früheren XForms Entwurf wurde diese Form auch Button genannt. Schließlich wurde sie aber auf den jetzigen Namen geändert, da ein Button nur eine Möglichkeit von vielen ist, die mit „trigger“ realisiert werden kann. Andere Möglichkeiten wären unter anderem Bilder (Images), Hyperlinks, Stimmaktivierungen, etc. Im Anhang ein Beispielcode für „trigger“. [41]

```
<trigger>
  <label>Login</label>
  ...
</trigger>
```

- submit

  „Submit“ ist ein Spezialfall von „trigger“ mit dem Effekt, dass die Daten an das angegebene Ziel übertragen werden. Hierfür ist ein Submission-Element im Model notwendig, welches im untenstehenden Beispiel ersichtlich ist. [41]

```
// Submission in Model
<xforms:submission action="http://<Ziel>" method="post" id="submission"/>
// Submit
<xforms:submit submission="submission">
  <xforms:label>Submit</xforms:label>
</xforms:submit>
```

- select1

  Mit dieser Form ist es möglich Selektionen aus einer Liste vorzunehmen. Dabei gibt es nur eine Auswahlmöglichkeit. Die Form selbst sagt dabei noch nichts über ihre genaue Darstellung aus, dafür gibt es das Attribut „appearance“. Der Wert „full“, welcher im Beispiel Verwendung findet, lässt die Form in einer Liste von Checkboxen oder Radio Buttons anzeigen. Mit dem Wert „compact“ kann man eine kompaktere Liste mit Scrollbars erstellen. Und mit dem Wert „minimal“ lässt sich ein einziges Listenelement darstellen, wie z. B.: ein Drop-Down Menü. Nach dem gezeigten Beispiel folgt noch die grafische Umsetzung in einem Browser (Abbildung 18). [41]

```
<select1 ref="cpu" appearance="full">
  <label>CPU</label>
  <item>
    <label>Pentium 4 2.53Ghz - $220</label>
    <value>2.5</value>
  </item>
  <item>
    <label>Pentium 4 2.8Ghz - $415</label>
    <value>2.8</value>
  </item>
  <item>
    <label>Pentium 4 3.06Ghz - $620</label>
    <value>3.0</value>
  </item>
</select1>
```

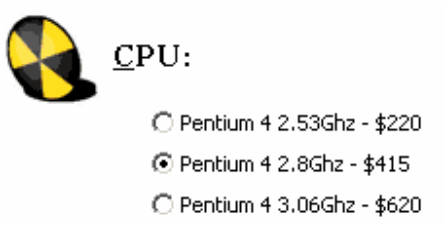

Abbildung 18: Select1 Form mit Appearance = full [41]

- select

  Wie bei „select1“ wird auch diese Form als Liste repräsentiert. Mit dem Unterschied, dass hier keine, eine oder mehrere Selektionen möglich sind. Ebenso ist das „appearance“-Attribut verwendbar. Die Darstellung ist wie bei der „select1“ Form. Anschließend ein Beispiel mit der visuellen Darstellung (Abbildung 19) in einem Browser. [41]

```
<select ref="cctype">
  <label>List For Specifying All Card Types</label>
  <item>
    <label>Master Card</label>
    <value>MC</value>
  </item>
  <item>
    <label>Visa Card</label>
    <value>VI</value>
  </item>
  <item>
    <label>American Express</label>
    <value>AE</value>
  </item>
  <item>
    <label>Diners Club</label>
    <value>DC</value>
  </item>
</select>
```

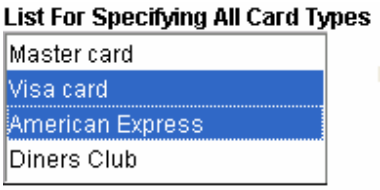

Abbildung 19: Select Form [41]

- label

  Das „label“ ist bereits bei den bisherigen Beispielen vorgekommen. Es wird verwendet, um Texte anzuzeigen. Es können aber auch Bilder (Images) eingebunden werden. [41]

- help, hint, alert

  Diese Formen werden verwendet, um Anwender Hilfestellungen zu geben oder sie auf Fehler, z. B. durch einen falschen Datentyp, aufmerksam zu machen. [41]

4.5.3 Validierung in XForms

Die Validierung sollte ein wesentlicher Bestandteil in jeder Sprache sein. Mit ihr lassen sich Daten auf Tippfehler, aber auch auf provozierte Fehler, überprüfen. In XForms sind hier drei wesentliche Faktoren maßgeblich, die zur Validierung beitragen, wie nachstehendes Binding mit CSS-Beispiel zeigt:

```
// Binding with contraint, relevant and type
<xforms:bind nodeset="<nodeset>" constraint="<constraint>"
                                    relevant= "<relevant>" type="<type>"/>

// Cascading style sheet for red visualization in case of error
*:invalid {
  color: red;
  background-color: transparent;
}
```

- Datentypen (types)

  Durch die im XForms-Prozessor fix eingebauten Datentypen (Built-in Types) sowie durch die in einem Schema erweiterten Typen ist eine Überprüfung des Typs in einem Eingabefeld möglich. Wenn also in einem Eingabefeld ein Typ „double“ erwartet wird und Buchstaben (string) eingegeben wurden, kann man solange keine Formdaten absenden, bis der Fehler korrigiert worden ist. Daher ist es nützlich „Not Used“ Werte einzuführen, die als Default-Werte fungieren. Im obigen Beispiel wäre z. B. „-1“ möglich. Nun würde der Parser erfolgreich validieren können und die Formdaten absenden lassen. In diesem Fall muss nur das Zielprogramm dahingehend angepasst werden, dass der gesendete Wert „-1“ kein Eingabewert eines Anwenders ist, sondern nur ein Default-Wert, welcher ignoriert werden kann. Die Visualisierung eines Fehlers in einem Eingabefeld zeigt sich anhand der definierten Cascading-Style-Sheet Definition, z. B. in roter Farbe.

- Limitierung (constraint)

  Jeder Typ lässt sich zusätzlich Einschränken. So kann ein Datentyp „double“ z. B. für alle Zahlen größer „0“ und kleiner „20“ limitiert werden: constraint="number(.) > '0' and number(.) < '20'". Die Verwendung von Entitäten „>“ und „<“ ist in Kapitel 2.1.1 nachzulesen. Soll außerdem ein “Not Used” Wert möglich sein, so muss eben gezeigtes contraint-Attribut um folgenden Befehl erweitert werden: "or number(.) = '-1'". Das bedeutet nun, dass alle Werte zwischen „0“ und „20“, inklusive dem Wert „-1“ gültig sind.

- Relevante Daten (relevant)

  Oft kann es nützlich sein, Eingabefelder nur dann anzuzeigen, sofern gewisse Kriterien erfüllt sind. Dadurch sind erst gar nicht falsche Eingabewerte möglich. Will man z. B. Operatoreingaben (<, >, <=, =>, =, !=) für ein Eingabefeld anzeigen lassen, so macht das nur dann Sinn, wenn das Feld nicht als „Not Used“ gefüllt ist: relevant="number(.) != '-1'".

4.5.4 Client- und server-seitige Verwendung von XForms

Bei client-seitigen Anwendungen ist für die Interpretierung von XForms das Endgerät (Browser, Handy, PDA, …) verantwortlich und bei server-seitigem Einsatz der Server bzw. die darauf laufende Applikation wie etwa Chiba [48] oder der Orbeon PresentaionServer [49].

Beide Arten haben ihre Vor- und Nachteile. Während bei ersterem Ansatz immer der Anwender für die aktuelle XForms Implementierung zuständig ist, sei es in dem er geeignete Plug-ins für seine Browser lädt, oder z. B. ein Software-Update an seinem Handy vornimmt, so ist bei letzterem Ansatz einzig und allein der Betreiber zuständig. Kann XForms an den Erfolg von HTML anschließen, so würde es natürlich ausreichen, die Verbreitung von XForms client-seitig zu forcieren. Da dies aber momentan noch nicht der Fall ist, gibt es eben auch Überlegungen in server-seitiger Verbreitung von XForms. Die server-seitige Architektur ist in Abbildung 20 ersichtlich. Eine XForms Engine ist notwendig, um den XForms Code in eine Sprache ((X)HTML, JavaScript und CSS) umzuwandeln (1), damit das Endgerät mit den Daten letztendlich was anfangen kann (2). Werden Daten vom Anwender an den Server geschickt (3), benötigt man wiederum die XForms Engine, welche die Daten rückwandelt (4).

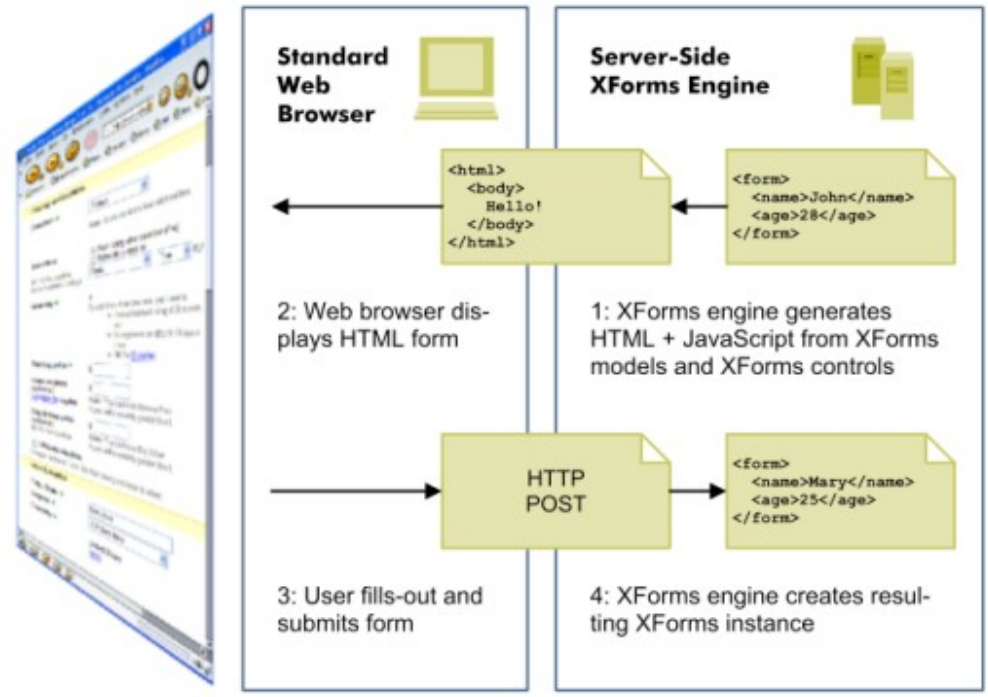

Abbildung 20: Server-seitige Architektur [54]

4.5.5 Installation von XForms

Hier wird nun auf die Komponenten eingegangen, welche benötigt werden, um XForms einzusetzen.

- Client-seitige Installation

  Zurzeit ist es nur möglich, XForms in Browsern darzustellen. Andere Endgeräte, wie z. B. das Handy, unterstützen XForms bislang nicht. Hier können z. B. nur etwaige XSL-Transformationen Abhilfe schaffen. Mit XHTML 2.0 [55] soll XForms standardmäßíg inkludiert sein. Bis aber der XHTML Entwurf zu einer Empfehlung und schließlich weite Verbreitung finden wird, muss man mit diversen Plug-ins in Browsern das Ausreichen finden. Die gängigsten Plug-ins für Browser sind:

  - Novell Plug-in für Internet Explorer 6.0 [56]
  - FormsPlayer Plug-in für Internet Explorer 6.0 [58]
  - Mozilla Extension für alle Mozilla Projekte (Firefox, SeaMonkey, XULRunner, etc.) [57]

  Die Plug-ins für den Internet Explorer sind einfach per Doppelklick auf die Installationsdatei zu installieren. Bei Mozilla ist die Installationsdatei eine Mozilla-Extension und daher auch als solche zu installieren. Für weitere Informationen über Extensions siehe auch https://addons.mozilla.org/extensions.php.

- Server-seitige Installation

  Zwei Projekte, die sich mit server-seitiger XForms Implementierung beschäftigen, sind wie schon erwähnt:

  - Chiba [48]
  - Orbeon PresentaionServer [49]

  Die Installationsanweisungen sind auf den jeweiligen Projekt-Homepages zu erfahren. Nach erfolgreicher Installation sind noch eventuelle Pfadangaben im XForms Code anzupassen. Gemeint ist hier der Sourcepfad zum Cascading Stylesheet, zum Logo (Image), zum Schema und zur externen Instanzdatei, wenn diese Daten extern ausgelagert sind.

Zusätzlich ist noch für beide Varianten zu beachten, dass die XHTML Namespaces unterschiedlich definiert sein müssen. Während Chiba den XHTML2 Namespace unbedingt benötigt, funktioniert die Extension von Mozilla und der Orbeon PresentationServer damit nicht. Diese benötigen wiederum den XHTML 1.0 Namespace. Das Novell- und das FormsPlayer - Plug-in kommen mit beiden Namespaces klar.

4.6 Generierung von XForms

Der Generierungsvorgang selbst ist ebenfalls von wesentlicher Bedeutung. Wie realisiert man aber nun am besten einen Code Generator? Aus einer Fülle von Möglichkeiten muss man zuerst eine Generator Engine finden, die den Anforderungen am besten entspricht. In dieser Arbeit wird mit Hilfe von Abbildung 21 die Engine ermittelt. Man beginnt beim Start-Punkt und anhand der jeweiligen zu generierenden Daten wählt man dann den entsprechenden Pfad. Da es sich bei der Sprache des Web-Interfaces um eine Markup-Sprache handeln wird (siehe Kapitel 4.3), wird die Auswahl auf „Custom code“ eingeschränkt, da weder Java Code oder ähnliches generiert werden soll. Somit verbleiben einige wenige vorgeschlagene Generator Engines die nun näher beleuchtet werden.

- Jostraca

  Jostraca ist ein in Java geschriebenes Entwicklungstool zur Code Generierung. Es erleichtert die Code Generierung durch ein Framework, welches Quellcode-Templates mit Meta-Daten vereinigt. Der Prozess ist ähnlich wie der von Java Server Pages. Obwohl Jostraca in Java geschrieben ist, kann man mit irgendeiner Programmiersprache Code für irgendeine andere Programmiersprache generieren. Jostraca entspricht dem Tier Generator Model. [17]

- GSL

  GSLgen ist ein Universalwerkzeug zur Datengenerierung und ist in C geschrieben. Es generiert Quellcode, Daten oder andere Dateien von einer XML Datei und einem Schema. Die XML Datei definiert Meta-Daten und das Schema wird dazu verwendet, um GSLgen mitzuteilen wofür diese Daten verwendet werden. GSLgen basiert auf zwei Konzepten. Das erste beinhaltet die Beschreibung von Daten-Objekten durch XML, da XML eine moderne und selbsterklärende Sprache ist. Das zweite Konzept inkludiert eine Schemasprache (GSL), mit welcher jede Art von Manipulation der XML Daten und der daraus resultierenden Ausgabe durchgeführt werden kann. GSL basiert auf dem Tier Generator Model. [18]

- Velocity

  Velocity ist eine Java basierende Template Engine nach dem Tier Generator Model. Wenn Velocity für Web-Entwicklungen verwendet wird, können Web-Designer parallel mit Java Programmierer gemäß dem Model-View-Controller (MVC) Modell arbeiten, was bedeutet, dass sich Web-Designer nur auf die Erstellung der Web-Seiten kümmern und dass auch die Programmierer sich nur auf erstklassige Programmierung fokussieren müssen. Velocity trennt nämlich Java Code von Web-Seiten und macht damit Web-Anwendungen über ihre Lebensdauer wartbarer und ist somit eine Alternative zu JSP und PHP. Velocity kann aber auch dazu verwendet werden, um SQL, PostScript oder XML Daten von Templates zu generieren. Es kann entweder als selbständiges Tool oder als integrierte Komponente für andere Systeme verwendet werden. [19]

- Nvelocity

  Velocity wird, wie vorher bereits erwähnt, hauptsächlich für die Generierung von MVC Frameworks verwendet. Nvelocity ist eine Erweiterung zu Velocity mit Nvelocity Templates in einem .NET MVC Framework zur Generierung von .NET Code und wurde implementiert in C#. Es entspricht ebenfalls dem Tier Generator Model. [20]

- CodeSmith

  Hier handelt es sich um eine in C entwickelte Implementierung nach dem Tier Generator Model. CodeSmith ist ein template-basierender Code Generator, welcher Code Generierung für jede Textsprache erlaubt, wobei durch entsprechende veränderbare Eigenschaften (Properties) die Code Generierung angepasst werden kann. So ein Property kann ein .NET Objekt oder ein einfaches boolesches Property sein. CodeSmith enthält viele Standard-Property Typen und ist zu 100% erweiterbar, indem sich der Anwender seine eigenen Property Typen kreiert. Die Syntax ist fast identisch mit ASP.NET. Wenn man also mit ASP.NET vertraut ist, sollte es kein Problem sein, diese Syntax schnell zu verstehen. C#, VB.NET oder JScript.NET Sprachen können in den Templates verwendet werden und jedes Template kann ASCII basierende Sprache ausgeben. [21]

- MetaL

  MetaL, eine dem Tier Generator Model entsprechende PHP Entwicklung, ist wie der Name schon sagt, eine Meta-Programmierungssprache. Meta-Programmierung ist eine Methode um Computer Programme zu erstellen. MetaL erwartet als Eingabedaten XML basierenden Quellcode. Eine MetaL Compiler Engine kann verwendet werden, um dasselbe Programm aus einem MetaL Quellcode in eventuell jede Zielsprache zu generieren. Momentan unterstützte Zielsprachen sind Perl, Java und PHP. [22]

- MetaStorage

  MetaStorage ist eine Persistence Layer Generator Anwendung basierend auf den Modulen von der MetaL Compiler Engine. Es ist geeignet zur Generierung von Objektklassen, welche eine Persistence Layer API von einem Modell, in XML definiertem Format, implementieren. Dieses Modell wird Component Persistance Markup Language genannt (CPML). Es entspricht ebenfalls dem Tier Generator Model. [23]

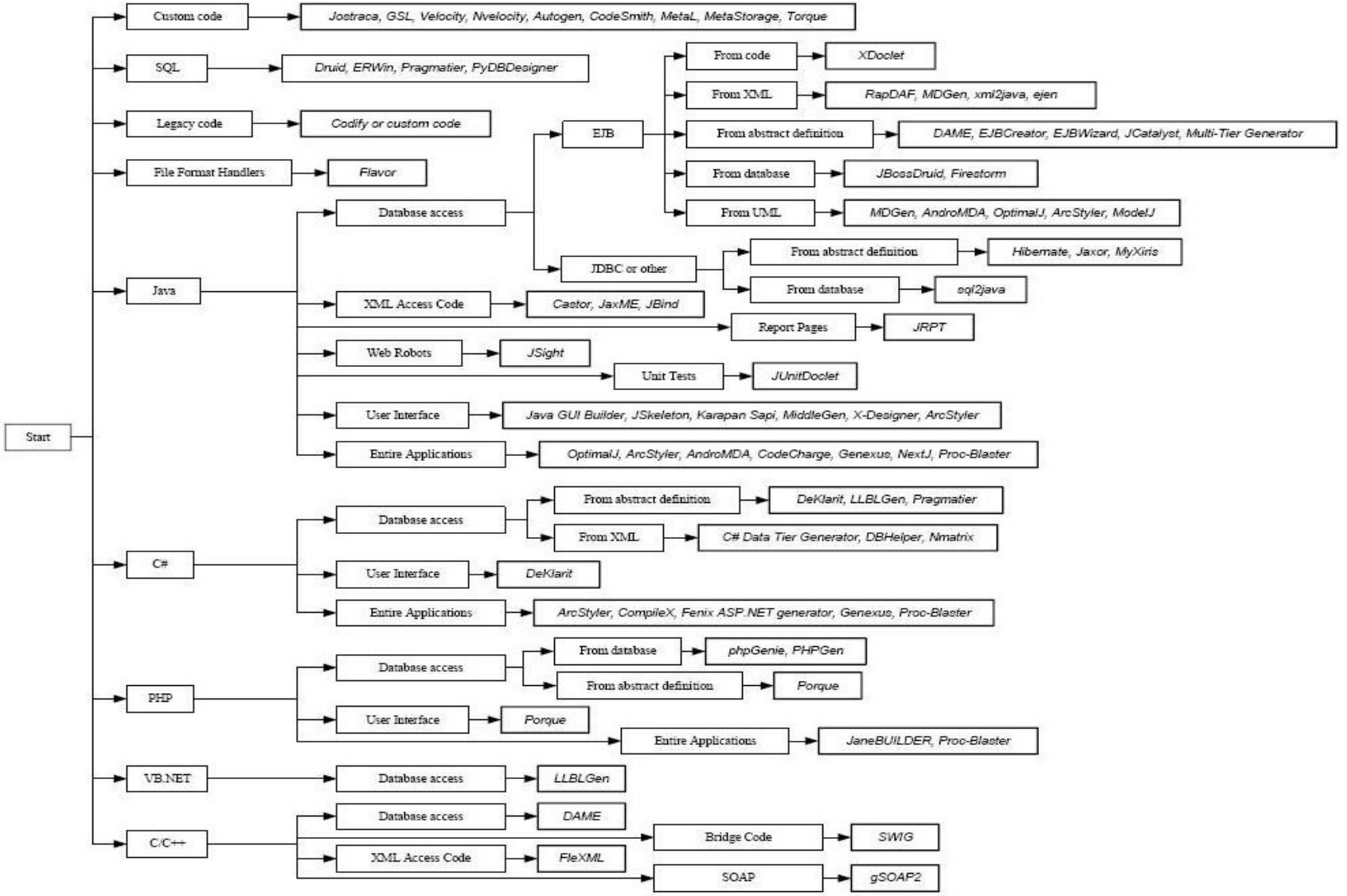

Abbildung 21: Code Generation – Entscheidungsbaum [16]

- Torque

  Torque, ebenfalls dem Tier Generator Model entsprechend, ist ein objekt-relationaler Mapper für Java. In anderen Worten, Torque gewährt Zugriff und erlaubt das Manipulieren von Daten in einer relationalen Datenbank, welche Java Objekte verwendet. Torque verwendet nicht wie andere objekt-relationale Mapper Java Reflection, sondern verwendet ein XML Schema, welches das Datenbank Layout beschreibt. Das XML Schema kann ebenfalls für die Generierung und Ausführung eines SQL Skripts verwendet werden, welches alle Tables in einer Datenbank anlegt. Ursprünglich war Torque als ein Teil von Turbine Framework [24] entwickelt worden, ist aber seit der Version 2.2 von Turbine entkoppelt. [25]

Es gibt noch viele Generatoren mehr, die ebenfalls Potenzial hätten und für diese Aufgabe verwendbar wären, aber hier nicht aufscheinen. Das heißt nicht, dass diese weniger geeignet wären, es handelt sich bloß um eine Auswahlentscheidung nach Abbildung 21. Die Generierung eines Markup-Textes ist das mindeste was ein Generator schafft, somit könnte jeder verwendet werden. Performance ist natürlich immer ein wichtiger Punkt, da aber der wesentliche Teil der Code Generierung nur einmal passiert, ein Anwender wie zum Beispiel ein Administrator wählt XPaths aus und daraus soll letztendlich ein dauerhaftes Web-Interface entstehen, ist dieser Punkt für die Betrachtung unkritisch und kann bei der Entscheidungsfindung außer Acht gelassen werden. Der Generierungsaufwand für eventuelle XUpdate Statements (Update, Delete, Insert, Append) während der Laufzeit wird als gering erachtet und kann ebenfalls für die Performanceentscheidung vernachlässigt werden.

Viel wichtiger ist hier, dass es sich um einen template-basierenden Generator handelt. Der Grund liegt darin, dass mit Templates Coding Styles bzw. Coding Guidelines, von Firma zu Firma unterschiedlich genannt, eingehalten werden können. Man kann den Code so schreiben, als wäre dieser handgeschrieben. Das Einfügen von Leerzeichen ist ebenso möglich, wie das Positionieren des Codes an beliebiger Stelle. [26] Das nächste Kriterium ist, dass Java die Implementierungssprache sein soll, da das gesamte Projekt mit Java realisiert wird und so auch alle einzelnen Komponenten. Diese Auswahl soll somit eine einfachere Integration des Generators in das Projekt mit sich bringen.

Mit diesen beiden Kriterien bleiben nur noch zwei Code Generatoren übrig (Velocity und Jostraca), da Torque zwar in Java geschrieben ist, aber nur für relationale Datenbanken ausgelegt ist. Welcher der beiden nun der bessere ist lässt sich gar nicht ermitteln. Auf den Homepages dieser Generatoren wird man natürlich nichts Negatives finden und diverse Vergleichstests gibt es zurzeit ebenfalls nicht. Außerdem haben beide ähnliche Regeln für die Template-Gestaltung, was die Entscheidung nicht einfacher macht. Da aber Velocity für Webentwicklungen gedacht ist, fällt letztendlich die Entscheidung auf diesen Generator.

4.7 Genereller Aufbau

Nachdem bereits alle wichtigen Details bekannt sind, kommt nun ein weiterer wichtiger Punkt in Kapitel Entwurf – ein ungefährer Umriss der Implementierungskomponenten, wie Abbildung 22 zeigt.

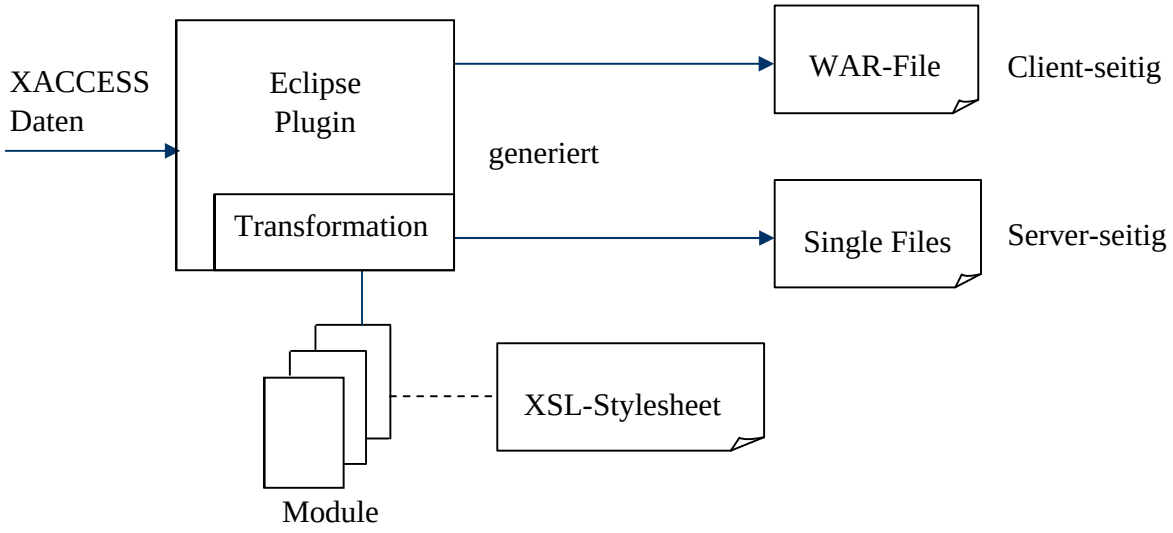

Abbildung 22: Aufbau

Wie schon vorher erwähnt, soll der Generator als Eclipse-Plugin realisiert werden. Diesem Plugin stehen die XAccess-Daten (aus XAccess-Datei) als Input zur Verfügung, welche alle möglichen XPaths und Restriktionen beinhaltet. Weiters soll dem Anwender die Möglichkeit offeriert werden, zwischen der Generierung einer WAR-Datei und einer Einzeldatengenerierung zu entscheiden. Die WAR-Datei wird bekanntlich für Web-Server (z. B.: Tomcat) verwendet und die hier bezeichnete Einzeldatengenerierung soll nichts anderes sein, als dass die sonst in einer WAR-Datei komprimierten Dateien, unkomprimiert in ein Zielverzeichnis generiert werden. So können sie in server-seitigen Lösungen wie Chiba [48] oder Orbeon [49] eingesetzt werden. Die etwaigen Konfigurationen für diese Anwendungen werden hier nicht berücksichtigt. Es wird lediglich die Möglichkeit angeboten, die generierten XForms-Dateien in der Einzeldatengenerierung über einem ausgewählten Modul mit seinem zugehörigen XSL-Stylesheet zu transformieren. Jedes so genannte Modul verkörpert ein Gerät (z. B.: Browser, Handy, PDA, …). Eine nähere Beschreibung über die Module findet man in Kapitel 5.3.1. Die Transformation ist notwendig, um den generierten XForms Source in das bestehende Dokument der Zielsprache einzubetten, da dies durch eine Trennung der Daten und der Logik von der Präsentation notwendig gemacht wird.

Die client-seitige Variante wird versucht über Servlets zu lösen, da die ausgewählten Selektionen im Admin-GUI in XPath- bzw. XUpdate-Statements anschließend noch verständlich zusammengebaut werden müssen, um eine nachher angesteuerte Datenbank auch sinnvoll abfragen zu können. Des Weiteren muss für die verschiedenen Endgeräte eine XSL-Transformation durchgeführt werden, die ebenfalls in einem eigenen Servlet passieren soll. Man kann hier also nicht von einer rein client-seitigen Lösung sprechen. Das XForms Formular inklusive der Validierung wird am Client durchgeführt und das Zusammensetzen der XPath- bzw. XUpdate-Statements mit einer Nachvalidierung geschieht dann im Servlet, also am Server. Trotzdem wird hier im Folgenden immer von einer client-seitigen Lösung gesprochen.

5 Implementierung

Dieses Kapitel befasst sich mit den wesentlichen Gesichtspunkten der Implementierung. Einerseits wird hier das System näher beleuchtet, andererseits wird versucht einen tieferen Einblick in die Technik zu geben, bzw. das Kapitel Entwurf zu verfeinern.

5.1 Architektur

In Kapitel 2.3.3 wurde bereits die SemCrypt Architektur vorgestellt. Nun folgt in Abbildung 23 eine mögliche Konfiguration des Frameworks von SemCrypt mit dem Prototyp, wobei rechteckige Symbole die beteiligten Komponenten des Projekts und der ovale Körper ein externes Objekt darstellen sollen. Alle grau hinterlegten Rechtecke stellen generische Komponenten dar, welche zur Deployment-Zeit erzeugt und installiert werden können. [63]

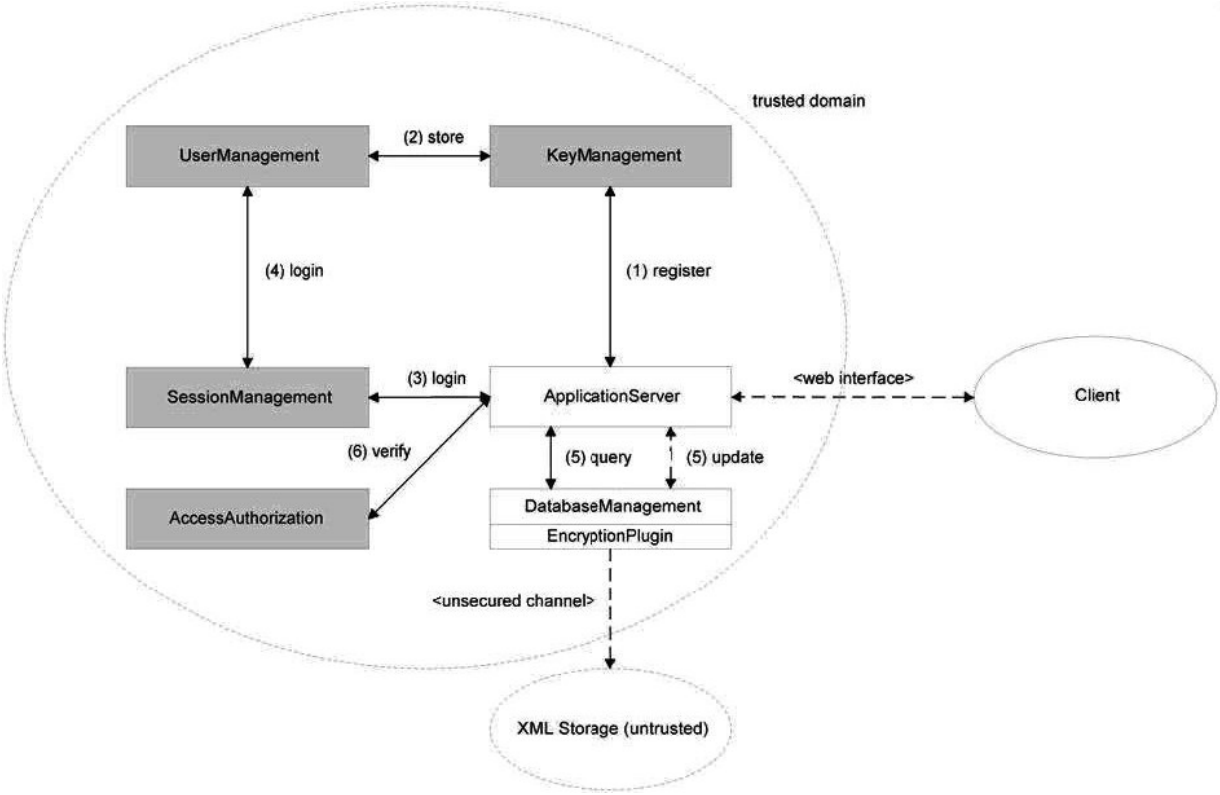

Abbildung 23: Architekturdesign [63]

Generell sieht der Ablauf folgendermaßen aus:

(1) Benutzer registriert sich über ein Web-Interface im System mit Username und Passwort.
(2) Benutzerdaten (Username und Passwort) werden gespeichert.
(3) Login ins System mit Benutzerdaten.

(4) Benutzerdaten werden kontrolliert, bei erfolgreicher Autorisierung werden dem User je nach Berechtigungsgrad mehr oder weniger Zugriffsmöglichkeiten gewährt.
(5) XPath / XUpdate Anweisungen werden geprüft (6) und anschließend an die unsichere Datenbank geschickt.

Kurzbeschreibung der beteiligten Komponenten:

KeyManagement:	Stellt Funktionalität für die Benutzererfassung bereit.
UserManagement:	Verantwortlich für Speicherung der Benutzerdaten.
ApplicationServer:	Koordinierung der Benutzeranfragen. Speicherort des Web-Interfaces.
SessionManagement:	Überprüft Benutzerdaten auf ihre Gültigkeit.
AccessAuthorization:	Beinhaltet die Lese- und Schreibberechtigungen der einzelnen Benutzerrollen.
DatabaseManagement:	Eng mit der Benutzeroberfläche gekoppelt. Der Zugriff auf die externe Datenbank erfolgt unverschlüsselt.

5.2 Eclipse Plug-in

Das Plug-in ist das Kernstück dieser Arbeit. Es ist verantwortlich mit den verfügbaren Daten aus der XAccess Datei (siehe Kapitel 4.1) das Web-Interface zu erstellen, mit welchem letztendlich Datenbankabfragen möglich sind. Im Folgenden wird das Kapitel in drei Bereiche unterteilt, um die wesentlichen Bestandteile des Plug-ins herauszuheben. Anschließend befinden sich noch wesentliche Details der Plug-in Konfiguration.

5.2.1 Wizard

Für die Erstellung des Plug-ins wird ein „Plug-in with a multipage-editor“ – Template verwendet. Ein Template ist nichts anderes, als ein vorgegebenes Plug-in, welches je nach Template gewisse Basisfunktionalitäten anbietet und sich somit als sehr hilfreich erweisen kann. Das ausgewählte Template stellt, wie der Name schon sagt, einen mehrseitigen Editor als Basisfunktionalität zur Verfügung, welcher das Gerüst für das Generator-GUI darstellt. Mit diesem können XPath-Selektionen je möglicher Benutzerrolle realisiert werden, indem man für jede Rolle eine eigene Seite verwendet, auf welcher die XPath-Auswahlmöglichkeiten dargestellt werden. Durch die möglichen Vor- und Rückwärtssprünge, die das Template ebenfalls bereitstellt, sind Korrekturen für den Anwender leicht möglich. Abbildung 24 zeigt das „multipage-editor“ Template nach Erstellung.

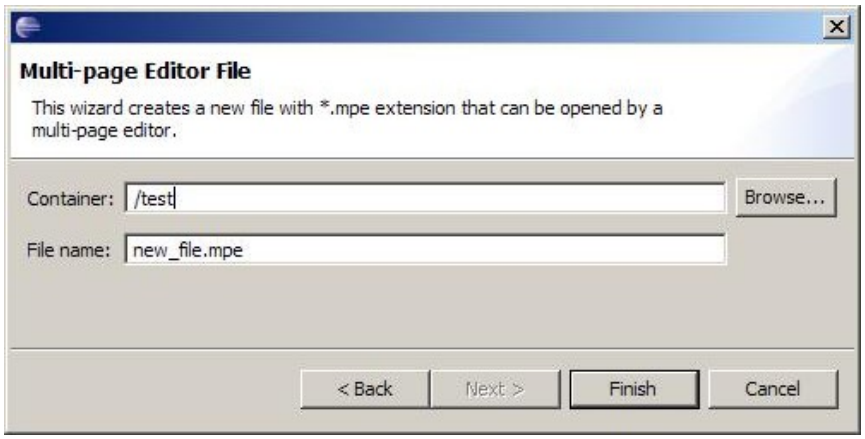

Abbildung 24: Multipage-Editor Template Plug-in

Neben den darzustellenden einzelnen Rollen pro Wizard-Seite, muss auch eine Seite mit diversen Konfigurationsabfragen inkludiert werden (WizardQuestionPage). Ziel dieser Seite soll sein, verpflichtende als auch optionale Daten vom Anwender zu erfragen, siehe dazu Abbildung 25.

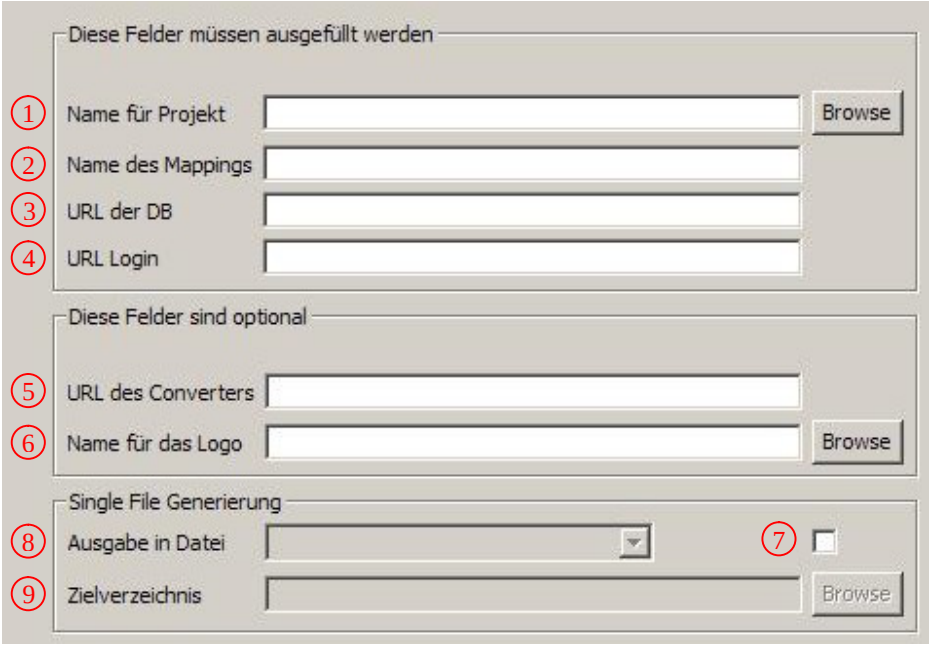

Abbildung 25: Ausschnitt der WizardQuestionPage

Das wären unter anderem der Name für die zu generierende WAR-Datei (1) und der Name des Mappings (2), welcher neben der Web-Server Adresse als URL und dem Namen der WAR-Datei ohne Dateityp zum Aufruf des Servlets führt: Z. B.: http://server:8080/semcrypt/MappingName. Letzterer wird dann in der „web.xml“ Datei des WAR-Archivs für das entsprechende Servlet in das URL-Pattern Tag im Servlet-Mapping Bereich, gespeichert:

```
<?xml version="1.0"?>
<web-app>
  <display-name>SemCryptRequester</display-name>
  <servlet>
    <servlet-name>MainServlet</servlet-name>
    <display-name>MainServlet</display-name>
    <servlet-class>at.ec3.semcrypt.servlet.MainServlet</servlet-class>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>MainServlet</servlet-name>
    <url-pattern>/MappingName</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

Die URLs für die Datenbankdestination (3), den Authentifizierungsserver für das Login (4) und dem Servlet (5), welches für die Nachvalidierung und den Zusammenbau der XPath- und XUpdate Statements verantwortlich ist, was bereits in Kapitel 4.7 angesprochen wurde, sind ebenfalls Teil dieser Seite. Die Möglichkeit, ein Logo für die zu generierende Seite zu laden, besteht hier ebenfalls (6). Weiters gibt es noch eine Auswahlmöglichkeit, ob ein WAR-Archiv- oder eine Einzeldatengenerierung durchgeführt werden soll (7). Bei der Einzeldatengenerierung müssen ein passendes Modul (8) und das Zielverzeichnis (9), wohin die Daten generiert werden sollen, angegeben werden. Die Beschreibung eines Moduls ist in Kapitel 5.3.1 zu finden. Diese Auswahl dient dazu, um für die Einzeldatengenerierung anzugeben, welches XSL-Skript für die Transformation herangezogen werden soll, das sich schließlich für die Einbettung der XForms Quelldaten in ein Zieldokument verantwortlich zeichnet.

Zu guter Letzt findet man noch einen „Finish“-Button, welcher die Generierung auslöst. Zuvor wird allerdings noch überprüft, ob für mindestens eine Rolle eine Auswahl getroffen wurde und ob der Name für das WAR-Archiv, der Name des Servlet-Mappings, die URLs für die Datenbankdestination und den Authentifizierungsserver bzw. die notwendigen Daten für die Einzeldatengenerierung vorhanden sind.

Nach einer allgemeinen Beschreibung des Wizards nun eine genauere Betrachtung des Geschehens. Über den Konstruktor des Plug-ins (SemCryptWizard.java) wird das Logging aktiviert, welches auf dem frei verfügbaren Log4j [51] aufbaut. Alle Fehlerfälle werden in einer Datei mitgeloggt. Außerdem wird versucht, ein ResourceBundle der jeweiligen Anwendersprache zu laden („wizard_<*Sprachcode*>.properties“), deren Inhalt verwendet wird, um die jeweiligen Beschreibungen, Überschriften, etc. im Plug-in in der richtigen Sprache anzeigen zu können.

Die „init“-Methode im WizardMaster (WizardMaster.java) ist danach zuständig, dass alle unterstützten Datentypen (Built-in Types wie string, double, boolean, …) von XForms und alle

frei definierbaren Datentypen in eine Hashtable geladen werden. Die Verwendung der Datentypen ist unter Kapitel 5.2.2 näher beschrieben. Während die Built-in Typen als Konstanten fix vorgegeben zu finden sind (IConstants.java), müssen alle frei definierbaren Typen zusätzlich in der Datei „types.xsd“ mit allen zugehörigen Schemata eingetragen werden. In XForms besteht so die Möglichkeit, jeden beliebigen Datentyp hinzuzufügen, wie zum Beispiel den Datentyp „AnyDateTimeType“, welcher in XForms nicht implementiert ist. Hierzu die Datei „types.xsd“ mit dem zusätzlich eingetragenen Typ:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <simpleType name="AnyDateTimeType">
    <union>
      <simpleType>
        <restriction base="dateTime">
          <pattern value="\d\d\d\d-\d\d-\d\dT\d\d:\d\d:\d\d(Z|(\+|-
                                                          )\d\d:\d\d)"/>
        </restriction>
      </simpleType>
      <simpleType>
        <restriction base="dateTime">
          <pattern value="\d\d\d\d-\d\d-\d\dT\d\d:\d\d:\d\d"/>
        </restriction>
      </simpleType>
      <simpleType>
        <restriction base="date">
          <pattern value="\d\d\d\d-\d\d-\d\d(Z|(\+|-)\d\d:\d\d)"/>
        </restriction>
      </simpleType>
      <simpleType>
        <restriction base="date">
          <pattern value="\d\d\d\d-\d\d-\d\d"/>
        </restriction>
      </simpleType>
    </union>
  </simpleType>
</schema>
```

In der Datei „types.properties“ muss der Name des neuen Datentyps mit dem „Not Used“ Wert und einem Validierungspattern für den Wizard eingegeben werden:

```
// Datentyp NotUsed-Wert Validierungspattern
// (Trennung erfolgt durch ein Leerzeichen, weitere Typen werden durch
//  Beistrich getrennt)
types=AnyDateTimeType 0001-01-01
                        (\\d\\d\\d\\d-\\d\\d-\\d\\dT\\d\\d:\\d\\d:\\d\\d) ...
```

Der “Not Used” Wert wird später für das Web-Interface benötigt, um Felder, welche nicht verwendet werden, auch als solche zu markieren. Standardmäßig ist jedes Feld auf „Not Used“ gesetzt. Die Einführung eines „Not Used“ Wertes ist notwendig, da der XForms Prozessor bei unausgefüllten Feldern eine Fehlermeldung wegen der Typenvalidierung ausgeben würde (siehe auch Kapitel 4.5.3). Das Validierungspattern ist für das in Abbildung 27 dargestellte GUI-Fenster notwendig. Beim Hinzufügen einer Restriktion wird gegen dieses Pattern validiert. Das Pattern für die Built-in Datentypen von XForms ist in „IConstants.java“ fix vordefiniert zu finden.

Weiters wird hier die Datei „xpath.xml“ eingelesen, die in Kapitel 4.1 als XAccess-Datei bereits vorgestellt wurde. Diese Datei definiert alle möglichen Zugriffspfade (XPaths), Benutzerrollen und deren Einschränkungen. Weitere Details zur Implementierung in Kapitel 5.2.2.

Zum Schluss werden hier nochmals sprachspezifische Daten eingelesen, die nun für die spätere Generierung des Web-Interfaces Verwendung finden und mit welchen pro vorhandener Sprache eine eigene Web-Interface Seite erstellt wird und alle textuellen Inhalte mit dieser Sprache ausfüllt. Für jede Sprache gibt es eine eigene Datei mit jeweils dem zugehörigen Sprachcode: „webinterface_<*Sprachcode*>.properties“. Für Österreich wäre das „de“, zugehörige Datei heißt dann folgendermaßen: „webinterface_de.properties“.

Nach diesen wichtigen Schritten wird im WizardMaster die Methode „addPages“ aufgerufen. Hier werden alle Seiten des Wizards registriert. Das sind je eine Seite pro Benutzerrolle (WizardTreePage.java) und die WizardQuestionPage (WizardQuestionPage.java), welche für die Konfigurationsabfragen verantwortlich ist, wie vorher bereits erwähnt wurde. Diese beiden Klassen besitzen jeweils die Methode „createControl“, die den GUI-Aufbau beinhaltet. In der WizrardQuestionPage sind die verwendeten GUI-Elemente das Button-, das Label- und das Text-Objekt. Die WizardTreePage stellt die möglichen XPath-Anweisungen (Elemente als auch Attribute) einer Rolle in einem Baum (Tree) dar. Die möglichen Zugriffspfade lassen sich visuell am einfachsten als Knoten (Node) im Baum darstellen. Wie das letztendlich in Eclipse dargestellt wird, zeigt der beispielhafte Baum mit Knoten für eine Rolle in Abbildung 26.

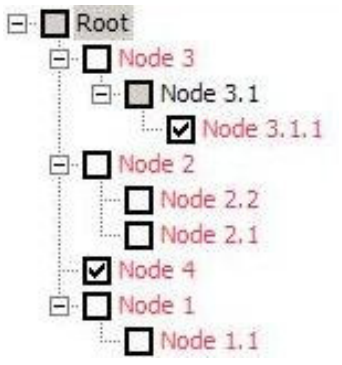

Abbildung 26: Baum mit mehreren Knoten für eine bestimmte Rolle

Der Knoten „Root“ befindet sich auf Ebene 0 und die Knoten 1 bis 4 auf Ebene 1. Knoten 3.1 auf Ebene 2 usw. Diese Definition wird noch später in Kapitel 5.2.3 benötigt. Weiters sind den Knoten Checkboxes vorangestellt. Eine aktivierte Checkbox bedeutet, dass dieser Knoten (mit all seinen Elternelementen entspricht dies einem XPath) für die angegebene Rolle erlaubt wird. In dem in Abbildung 26 gezeigten Beispiel wären das folgende zwei XPaths:

- /Root/Node 3/Node 3.1/Node 3.1.1
- /Root/Node 4

Das bedeutet nun, dass diese Auswahl grafisch in das Web-Interface umgewandelt werden muss, wo dann ein Anwender für die gegebene Rolle, die Elemente „Node 3.1.1“ und „Node 4“ aus der Datenbank erfragen oder diese mittels XUpdate modifizieren kann. Eine grau hinterlegte Checkbox bzw. schwarzer Text bedeutet, dass der Anwender keine Berechtigung besitzt, diese auszuwählen. Jedem Element bzw. Attribut können Einschränkungen vergeben werden. Dies geschieht durch einen Klick auf den jeweiligen Knoten. Abbildung 27 zeigt so ein Fenster. Hier werden alle möglichen Einschränkungen, die für das Knotenelement bzw. für das Attribut existieren sollen, eingetragen. Bestehende XPath-Restriktionen, die in der XAccess-Datei festgelegt sind, können nicht mehr aufgehoben werden, neue aber können nach Belieben hinzugefügt und auch wieder gelöscht werden. Bei den XUpdate-Restriktionen („Mögliche Rechte“) ist dies umgekehrt. Das Beispiel zeigt eine bereits eingetragene Einschränkung auf das Knotenelement, welche für „Node 3.1.1“ angibt, dass der Inhalt des Elements nicht (ne … not equal) Knoten heißen darf (Annahme: Node 3.1.1 ist ein String-Datentyp). Der Datentyp wird immer beim Hinzufügen einer neuen Restriktion, sei es nun ein Attribut oder Element, angezeigt (siehe Abbildung 28). Außerdem sind für das Element nur Leserechte vorhanden.

Abbildung 27: Restriktionsfenster

Abbildung 28: Eingabefenster mit Datentyp

Folgende Operatoren werden unterstützt und auch so eingetragen [59]:

- eq equal
- ne not equal
- lt less than
- le less equal
- gt greater than
- ge greater equal

5.2.2 XAccess Package

Dieses Package ist für das Einlesen und temporäre Zwischenspeichern der Daten zuständig, die in Abbildung 22 als XAccess-Daten benannt, dem Plug-in zugeführt werden. Wie bereits in Kapitel 2.1.6 erwähnt, steht eine Schema-Datei „xaccess.xsd“ zur Verfügung, die beschreibt, wie die XAccess-Datei „xpath.xml“ auszusehen hat. Für das Einlesen der Daten aus der XAccess-Datei, wie z. B.: die einzelnen Rollen oder die möglichen Rechte wie das Löschen, Einfügen oder Aktualisieren, bedient man sich der XMLBeans-Technologie [52]. Aus dem Schema wird eine Archiv-Datei generiert (XMLBeans-Befehl für Generierung: scomp –out <archiv.jar> <schema.xsd>), mit welcher man auf die einzelnen Elemente und Attribute problemlos zugreifen kann:

```
// Zugriff auf Wurzel-Element
XAccessType xatype = XAccessDocument.Factory.parse(<XML file>).getXAccess();
// Alle Rechte (insert, delete, ...) werden eingelesen
SecurityRightType sec_right_type[] =
      xatype.getSecurityRights().getSecurityRightArray();
// Alle XPath-Möglichkeiten (mit Datentyp) werden eingelesen
SecurityPathType sec_path_type[] =
      xatype.getSecurityPaths().getSecurityPathArray();
// Alle Restriktionen werden eingelesen
PermissionType permType[] = xatype.getPermissions().getPermissionArray();
```

Aus den eingelesenen Daten wird nun für jeden XPath mit seinen Einschränkungen und Berechtigungen jeweils ein XPath Objekt (XPath.java) erzeugt. Die zugehörigen Restriktionen für den XPath sind ebenfalls in eigenen Objekten im XPath Objekt enthalten (RestrictionPath.java).

Während des Auslesens der einzelnen SecurityPaths wird auch der Datentyp mitgeliefert. Es können hier bekannte Typen wie „string“, aber auch unbekannte wie „SemCrypt_CompetenciesType“ vorkommen. Manche Elemente besitzen überhaupt keine Definition, wie z. B. Competency. Entspricht der ausgelesene Datentyp weder einem Built-in Datentyp noch einem Selbstdefinierten, so wird er auf „anonymous“ gesetzt und bei der späteren Generierung ignoriert. Das heißt, dass z. B. das Element Competency kein Input-Feld im GUI besitzt, aber ein Element SubscriberNumber vom Typ „string“ sehr wohl, sofern dafür

entsprechende Rechte vergeben wurden. Das Admin-GUI spiegelt das gleiche Verhalten wider. Für Elemente ohne bekannten Datentyp können keine Restriktionen vergeben werden.

Für die Permissions ist wichtig, dass bei mehrmals gleich vorkommenden PathValues immer der gesamte Permission-Block neu eingetragen werden muss. Haben ein Projektleiter und ein Gruppenleiter beide gemeinsame Rechte (z. B. Leserechte), so ist es nicht möglich, diese gemeinsam in einem Block einzutragen und in einem neuen nur mehr jene Rechte, welche sich zur anderen Rolle unterscheiden:

```
// Trotz Gemeinsamkeiten Aufsplittung notwendig
<PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
<RoleName>GL</RoleName>
<RightName>read</RightName>
...
<PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
<RoleName>PL</RoleName>
<RightName>read</RightName>
<RightName>delete</RightName>

// Leserechte in einem gemeinsamen Block für PL und GL sind nicht möglich
<PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
<RoleName>GL</RoleName>
<RoleName>PL</RoleName>
<RightName>read</RightName>
...
<PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
<RoleName>PL</RoleName>
<RightName>delete</RightName>
```

Aufgrund der Verwendung von Hashtables, welche in der Implementierung als Key den XPath haben, ist dies nicht möglich, da sie sich sonst bei gleichen Rollen überschreiben würden.

Des Weiteren befindet sich in dem Package noch die Klasse UserRole (UserRole.java), welche die Information über alle Rollen beinhaltet. Außerdem sind über diese Klasse alle XPaths für eine bestimmte Rolle abrufbar, was für die spätere Generierung im Generator notwendig ist.

5.2.3 Generator Package

Hier befinden sich die, für die Generierung des Web-Interfaces, wichtigsten Dateien. Zum einem der Parser (Parser.java), welcher die vorhandenen XPath Objekte parst und der Generator, der schließlich die GUI-Dateien generiert.

Um den Parser und den Generator verstehen zu können, muss vorerst der Aufbau des Web-Interfaces näher beleuchtet werden. Die grundlegende Idee ist, eine XForms-Seite zu erstellen und die veränderbaren Textelemente und Attribute, welche letztendlich generisch erstellt werden sollen, durch den Befehlssatz der Velocity Engine zu ersetzen.

- Aufbau des Web-Interfaces

  Wie bereits in Kapitel 4.4 erwähnt wurde, fiel die Entscheidung auf XForms. Das heißt, dass das Web-Interface mit XForms, einem Nachfolger von HTML, aufgebaut wird. Der Aufbau der Seite kann grob in vier Bereiche eingeteilt werden. Der erste Bereich beinhaltet die Überschrift und ein Logo. Der zweite Bereich enthält ein Menü zur Ansteuerung der einzelnen Unterbereiche in SemCrypt (SemCryptEmployeeInfo, PersonInfo, Salary, JobPositionHistory, Competencies, AssessmentResults). Das sind alle Elementknoten der Ebene 0 und 1 (Ebenendefinition siehe Kapitel 5.2.1). Natürlich scheinen nur jene auf, die auch Kindelemente oder selbst keine Einschränkungen haben. Im dritten Bereich befindet sich die Navigation und im letzten Bereich die Darstellung der Elemente und Attribute. Im Navigationsbereich sind die unterschiedlichen XPaths anzuwählen. Der Aufbau ist wie im Wizard baumförmig, wobei hier die einzelnen Knoten als Radio Buttons dargestellt werden und durch das Anklicken eines Radio Buttons tiefer in den Baum vorgedrungen werden kann. Die generelle Gliederung sieht nun folgendermaßen aus:

  - Namespace (nicht generisch)
  - XForms:Model mit Instanzdaten für Navigation (nicht generisch)
  - XForms:Model mit Instanzdaten und Bindings für Submit (generisch)
  - XForms:Output für Logo (nicht generisch)
  - XForms:Label für Copyright (nicht generisch)
  - XForms:Label für Überschrift (generisch)
  - XForms:Trigger für Menüleiste (generisch)
  - XForms:Switch mit Select1 Anweisungen für Navigation (generisch)
  - XForms:Switch mit Select1 Anweisungen für Element- und Attributauswahl (generisch)
  - XForms:Submit Buttons (generisch)
  - XForms:Input Feld (generisch)

  Eben vorgestellte Gliederung entspricht immer dem Ergebnis des Generators. Teilgliederungen mit „nicht generisch“ werden immer so vom Generator generiert, wie sie im Velocity Template sind. Das heißt nichts anderes, dass jede Textzeile im Template 1:1 in die Output-Datei übernommen wird. Die übrigen enthalten im Velocity Template Referenzen auf die entsprechenden Objekte und werden je nach Auswahl im Wizard unterschiedlich generiert bzw. weggelassen. So eine Referenz ist nichts anderes als ein normaler Methodenaufruf auf ein Klassenobjekt nur eben zur Laufzeit. Besser bekannt auch als Java-Reflection. Hier ein Beispiel des Templates um es zu verdeutlichen: $naming.getHeadlineName(). Das Dollarsymbol zeigt dem Velocity Parser an, dass nun ein Aufruf erfolgt. Im Generator gibt es ein VelocityContext Objekt (inkludiert java.util.Map), in welches das Schlüssel/Werte Paar

mit dem Namen „naming“ als Schlüssel und dem Objekt mit der Methode „getHeadlineName()“ als Wert, gespeichert wird. Dadurch, dass die Methode nicht vom Typ „void“ sein kann, wird immer ein Wert zurückgeliefert und der Ausdruck „$naming.getHeadlineName()“ mit dem Ergebniswert der Methode ersetzt. Statt des Dollarsymbols gibt es auch noch das Rautesymbol, welches für „if/else“ und „foreach“ Anweisungen verwendet wird.

- Teilgliederungen im Detail

  - Namespace

    Hier muss unter anderem der Namespace für XForms deklariert werden (http://www.w3.org/2002/xforms). Des Weiteren besitzt XForms XML Events, die z. B.: durch das Drücken eines Buttons ausgelöst werden, diese benötigen ebenfalls einen eigenen Namespace (http://www.w3.org/2001/xml-events). Zu guter letzt ist noch ein eigener Namespace für die Datentypen notwendig (http://www.w3.org/2001/XMLSchema).

  - XForms:Model für Navigation

    Das Model (inklusive der Instanzdaten) wird nur für die Navigation im GUI genutzt. Die Daten dienen der aktuellen Sicherung der momentanen Position im Baum. Wird nun also mittels Radio Buttons eine tiefere Ebene angewählt, so wird dies dort zwischengespeichert.

```
<xforms:model id="navigation-data">
  <xforms:instance id="navigation-data-instance" xmlns="">
    <navigation>
      <tmp/>
    </navigation>
  </xforms:instance>
</xforms:model>
```

  - XForms:Model für Submit

    Dieses Model besitzt alle notwendigen Bindings auf die Instanzdaten, welche letztendlich an den Converter bzw. die Datenbank weitergeleitet werden. In diesen Instanzdaten werden also all jene Daten gespeichert, die vom Anwender im Web-Interface eingegeben werden und auch solche, die konstante Werte beinhalten. Anschließend das Model und die externe Instanz-XML-Datei:

```
// Model
<xforms:model
```

```
  #if($schema.booleanValue()) // Lade Schema bei zusätzlichen Datentypen
    schema="<schema>"
  #end
  id="<id>">
  // Lade Instanzdaten von externer XML Datei
  <xforms:instance src="<xml>" id="<id>" xmlns=""/>
  // Generiere alle notwendigen Bindings
  #foreach ($object in $objects)
    <xforms:bind nodeset="<nodeset>" constraint="<constraint>"
                        relevant="<relevant>" type="<type>" id="<id>"/>
  #end
  // Generiere alle notwendigen Buttons (insert, delete, ...)
  #foreach($rightname in $rightnames)
    #if($rightname.isRelevant())
      <xforms:bind nodeset="<nodeset>" relevant="<relevant>" id="<id>"/>
      <xforms:submission action="<action>" method="post" id="<id>">
        <xforms:action ev:event="xforms-submit">
          <xforms:setvalue ref="<ref>"><value></xforms:setvalue>
        </xforms:action>
      </xforms:submission>
    #end
  #end
  ...
</xforms:model>
// Externe Instanz-Daten
<data>
  <Selector mode="read"/>
  <InsApp/>
  #foreach($rightname in $rightnames)
    #if($rightname.isRelevant())
      <$rightname.getRightName() database="<database>"/>
    #end
  #end
  #foreach ($object in $objects)
    <$object.getName()_$object.getId() restriction="<restriction>"
           location="<location>" type="<type>" operator="eq" checked="">
      <value>
```

```
    </$object.getName()_$object.getId()>
  #end
</data>
```

Im Model wird dann ein Schema geladen, wenn ein zusätzlicher Datentyp in der Datei “types.xsd” vorhanden ist, also ein Datentyp der von XForms nicht unterstützt wird. Die Instanzdaten dieses Models befinden sich in einer eigenen XML-Datei und werden über das Attribut „src“ bezogen. Die Auslagerung der Instanzdaten in eine separate Datei hat den Vorteil, dass der Quelltext lesbarer wird und der Zugriff auf die ausgelagerten Daten für Dritte unterbunden werden kann. In den ForEach-Schleifen werden dann die restlichen Model-Daten eruiert, welche zuvor durch den Parser ermittelt wurden.

Die Instanzdaten nun genauer betrachtet, beginnen mit dem „Selector“, welcher aus einem ausgewählten Element mit Id (Verwendung von Element bzw. Id siehe später in diesem Kapitel) besteht. Nachstehendes Beispiel einer generierten Instanzdatei soll dies zeigen:

```
// Generiertes Instanzdatenbeispiel
<?xml version="1.0"?>
<data>
  <Selector mode="read">SemCryptEmployeeInfo_0</Selector>
  <read database="http://localhost:8888/db?xpath="/>
  <Competency_7 restriction="" location=".../Competency/Competency"
                type="anonymous" operator="eq" checked=""></Competency_7>
  <minValue_26 restriction=". eq '2006-10-10'"
    location=".../CompetencyEvidence/@minValue" type="string" checked="">
    Not Used</minValue_26>
  <SemCryptEmployeeInfo_0 restriction="" location="/SemCryptEmployeeInfo"
      type="anonymous" operator="eq" checked=""></SemCryptEmployeeInfo_0>
  ...
</data>
```

Ein „Selector“ mit der Information „SemCryptEmployeeInfo_0“ bedeutet, dass der Anwender im GUI dieses Element ausgewählt hat. Dieses Element ist außerdem in der Liste darunter enthalten. Der „mode = read“ wiederum besagt, dass es sich um eine XPath-Abfrage handelt, wobei weiters „update“, „delete“, „insert“ und „append“ möglich sind und jeweils den Abfragemodus angeben. Weiters ist für den ausgewählten Modus die Datenbankzieladresse enthalten. Anschließend sind alle Elemente und Attribute aufgelistet, die wie schon vorher erwähnt auch das „Selector“-Element beinhalten. Die Struktur ist immer dieselbe: <Element/Attribut + Id> <restriction> <location> <type> <operator> <checked> <value>. Das „Restriction“-Attribut besitzt die Einschränkungen, die bereits im Admin-GUI bzw. in der XAccess-Datei getroffen wurden, das „Location“-

Attribut den XPath, wo das Element/Attribut in der Dokumentstruktur gefunden werden kann. Der „type“ ist der Datentyp und der „operator“ die Auswahl des Operators im generierten GUI. Das „checked“-Attribut ist dann gesetzt, wenn im GUI eine Checkbox angeklickt wurde (benötigt für „insert“, „append“, „delete“ und „update“). Zu guter letzt noch der „value“ selbst, welcher entweder den „Not Used“ Wert inne hat oder den im GUI eingegebenen (bei „type = anonymous“ ist dieser nicht gefüllt).

- XForms:Output für Logo

Mit diesem Tag wird ein Logo (z. B.: SemCrypt.jpg) ausgegeben. Es liegt dann nur am jeweiligen Gerät (Browser, Palm, Handy, …) wie es damit umgeht.

```
// Logo
<xforms:output value="concat('&#60;img width=&#34;217&#34;
               height=&#34;60&#34; src=&#34;', '$image', '&#34; />')"/>
```

- XForms:Label für Copyright und Überschrift

Neben dem Logo wird ein Copyright-Datum ausgegeben und darunter eine Überschrift für die erstellte Seite. Der Text wird aus der Datei „webinterface_<*Sprachcode*>.properties“ bezogen (siehe auch Kapitel 5.2.1 – Erklärung der „init“-Methode), welcher dann von der Velocity Engine richtig eingesetzt wird.

```
// Copyright
<xforms:label class="copyright">&#169; 2006</xforms:label>
// Überschrift
<xforms:label class="headline">$naming.getHeadlineName()</xforms:label>
```

- XForms:Trigger für Menüleiste

Die Menüleiste dient der besseren Navigierbarkeit. Wie schon bereits erwähnt, ist das Menü generisch und von der Auswahl im Wizard und den Einschränkungen der XAccess-Daten abhängig (alle Elemente der Ebene 0 und 1). Die einzelnen Cases werden durch den Parser ermittelt.

```
#foreach($object in $objects)
  #if ($object.isMenuItem())
    <xforms:trigger class="button">
      <xforms:label>$object.getElement()</xforms:label>
      <xforms:reset ev:event="DOMActivate" model="navigation-data"/>
      <xforms:action ev:event="DOMActivate">
```

```
        <xforms:toggle case="<case x>"/>
        <xforms:toggle case="<case clear>"/>
      </xforms:action>
    </xforms:trigger>
  #end
#end
```

- XForms:Switch für Navigation

Die im Admin-GUI ausgewählten XPaths werden in diesem Bereich dargestellt. Jeder XPath wird durch einen Radio Button visualisiert. Durch das Auswählen eines eben genannten Buttons gelangt man weiter zu dessen Kindelementen. Diese Kindelemente sind ebenfalls wieder durch Radio Buttons auswählbar und können ihrerseits wieder Kindelemente haben, die wiederum auswählbar sind, usw. Um wieder eine Ebene höher zu kommen, gibt es zusätzlich einen Zurück-Button. Nachstehend eine überblicksmäßige Darstellung des Navigationsbereiches, wobei die „foreach“-Schleife nur dann durchgeführt wird, wenn das „$object“ kein Attribut ist und Kindelemente oder Attributelemente besitzt oder selbst ein auswählbarer Datentyp ist:

```
<xforms:switch ref="<reference>" model="<model>" id="<id>">
  #foreach($object in $objects)
    #if (!$object.isAttribute() && ($object.isChoosable() ||
                    $object.hasChildren() || $object.hasAttributes()))
      <xforms:case id="<id>" selected="false" class="internal-box">
        <xforms:select1 bind="<bind>" model="<model>" appearance="full">
          <xforms:item>
            <xforms:label><label></xforms:label>
            <xforms:value><value></xforms:value>
          </xforms:item>
        </xforms:select1>
        #if ($object.hasBackButton())
          <xforms:trigger class="back_button">
            <xforms:label>Back</xforms:label>
            <xforms:action ev:event="DOMActivate">
              <xforms:reset ref="<reference>"/>
              <xforms:toggle case="<case>"/>
            </xforms:action>
          </xforms:trigger>
        #end
```

```
      </xforms:case>
    #end
  #end
</xforms:switch>
```

- XForms:Switch für Attribut- und Elementauswahl

Der Aufbau ist ähnlich dem Navigationsbereich. Hier wird zu jedem im Navigationsbereich ausgewählten Element eine Liste aller möglichen Attribute angezeigt. Ist das Element selbst ebenfalls von einem bekannten Datentyp, so erscheint auch hierfür ein Feld für mögliche Eingaben. Jedes dargestellte Feld hat je nach Berechtigung eine Checkbox, welche für folgende Anwendungsfälle notwendig ist: Insert, Delete, Append, Update. Mit der ausgewählten Checkbox wird die Position gesetzt, an welcher z. B.: das Einfügen eines neuen Elementes oder das Löschen des Elements stattfinden soll. In Abbildung 29 ist ein Beispiel zu sehen, in welchem für das Attribut „dateOfIncident“ Lese- und Update-Rechte vorhanden sind. Der „Not Used“ Button wurde bereits mehrfach erwähnt und dient dazu ein Feld mit einem für den jeweiligen Datentyp gültigen Wert zu füllen, um die Typenvalidierung erfolgreich durchführen zu können. Das Attribut „expirationDate“ vom Typ „Date“ wurde bereits mit einem gültigen Datumswert befüllt, wodurch eine Choicebox sichtbar wird, die alle unterstützten Operatoren [59] auswählen lässt. Die Choicebox wird bei jedem Feld, welches Werte ungleich dem „Not Used“ Wert beinhaltet, angezeigt: Z. B. relevant="/data/expirationDate != '0001-01-01'" (nähere Details zu „relevant“ siehe Kapitel 4.5.3). Dies gilt gleichermaßen für Attribute als auch Elemente.

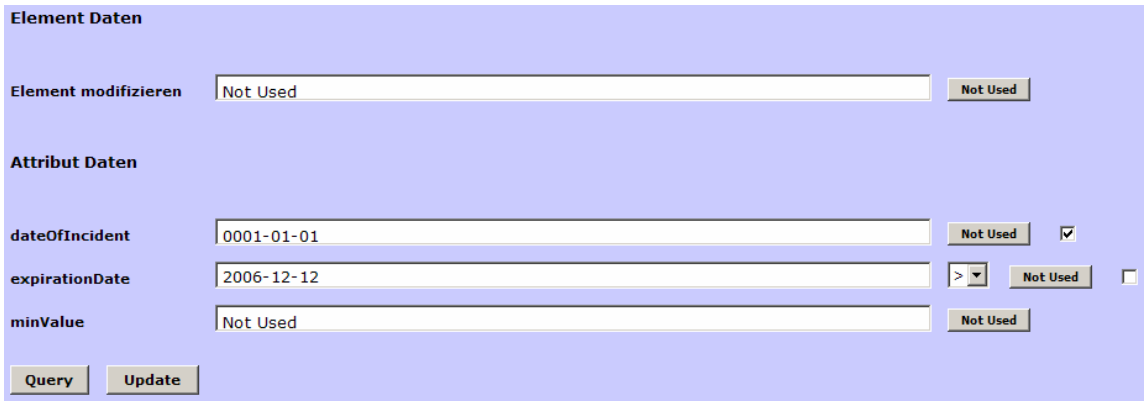

Abbildung 29: Attribut- und Elementauswahl mit Novell Plug-in auf IE6.0

- XForms:Submit Buttons

Die Submit-Buttons werden benötigt, um die Datenbankabfragen durchzuführen. Es gibt folgende Buttons, welche je nach Anwendungsfall generiert werden können: Insert, Delete, Append, Query, Update (siehe dazu auch Abbildung 29, welche Query und Update zeigt).

```
#foreach($rightname in $rightnames)
  #if($rightname.isRelevant())
    <xforms:submit model="<model>" bind="<bind>"
                        submission="<submission>" class="submit">
      <xforms:label><label></xforms:label>
    </xforms:submit>
  #end
#end
```

- XForms:Input Feld

Das Input-Feld ist dann rechts neben den Submit-Buttons sichtbar, wenn mindestens eine Checkbox eines beliebigen Elementes oder Attributes mit einer „insert“ oder „append“ Abfrage ausgewählt wurde. Das ist notwendig, um die zusätzlichen Daten (Name von Element/Attribut und Wert) vom Anwender zu erfragen. Siehe dazu das Beispiel in Abbildung 30, in welchem bereits der neue Elementname und sein Wert eingegeben wurden.

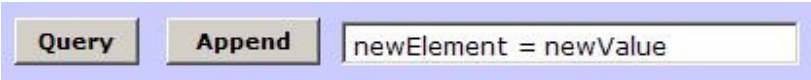

Abbildung 30: Input-Feld dargestellt mit Novell Plug-in auf IE6.0

Bei erfolgreichem Submit der Formulardaten mit Append würde das folgende XUpdate-Anweisung für das Beispiel PayrollInstructions durch den Converter liefern:

```
http://<server>:<port>?xupdate=
<?xml version="1.0" encoding="utf-8"?>
<xupdate:modifications version="1.0"
                      xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:append select="/.../PayrollInstructions">
    <xupdate:element name="newElement">newValue</xupdate:element>
  </xupdate:append>
</xupdate:modifications>
```

Zum Abschluss noch der notwendige Quellcodeauszug zur Generierung der Daten:

```
#if ($relevant.isVisible())
  <xforms:input model="<model>" class="input-insapp" bind="<bind>">
    <xforms:label class="space-insapp"/>
    <xforms:hint class="hint-string"><hint></xforms:hint>
```

```
  </xforms:input>
#end
```

Nachdem der Aufbau des Web-Interfaces erklärt wurde, kann nun auf den Parser und Generator näher eingegangen werden.

Die Aufgabe des Parsers besteht darin, die Daten, die in den Velocity-Templates benötigt werden, zu ermitteln und zur Verfügung zu stellen. Der Parser arbeitet in sechs Schritten. Im ersten wird durch alle bereits ermittelten XPath-Objekte (Erzeugung der XPath-Objekte in Kapitel 5.2.2 nachzulesen) durchiteriert. Dabei werden einige Informationen von den Elementobjekten die im Admin-GUI ausgewählt wurden, in einem neuen Datenobjekt gespeichert (DataObject.java). Die wichtigsten Informationen werden am Beispiel von CompetencyEvidence gezeigt:

XPath: /SemCryptEmployeeInfo/Comptency/Competencies/CompetencyEvidence

Attribute: @dateOfIncident
@name
@typeDescritpion
@expirationDate
…

Die wichtigsten gespeicherten Informationen und deren Wert für das Beispiel:

- XPath: /SemCryptEmployeeInfo/Comptency/Competencies/CompetencyEvidence
- Name: CompetencyEvidence
- Id: <fortlaufende Identifizierungszahl>
- Element: CompetencyEvidence
- ElementId: <fortlaufende Elementidentifizierungszahl>
- Datentyp: <jeweiliger Typ des Elements>
- NotUsed-String: Wert, mit welchem das Feld befüllt wird, wenn es nicht in Verwendung ist <siehe Kapitel 5.2.1>
- Ebene: 3 (Hierarchie im Baum mit 0 beginnend)
- Attribut: nein (Speicherung ob es sich um ein Attribut oder Element handelt)
- BackButton: ja (Speicherung ob ein Zurück-Button benötigt wird)
- Einschränkungen: es werden alle Restriktionen gespeichert
- Parent: Competencies (Elternelement des Objekts)
- Children: EvidenceId, … (Kinderelemente des Objekts)
- MenüItem: nein (Speicherung ob Objekt im Menü erscheinen soll)
- …

Im nächsten Schritt werden die Elternelemente von den im Admin-GUI ausgewählten XPath-Attributen ermittelt. Dabei handelt es sich um jene Elternelemente die im Admin-GUI nicht ausgewählt wurden, aber für die Zusammenstellung des Web-GUIs notwendig sind. Hier werden ebenfalls wieder dieselben Informationen in einem Datenobjekt gespeichert wie im ersten Schritt,

nur wird das Element als nicht ausgewählt markiert. Im dritten Schritt werden alle Informationen der gefundenen Attribute abgespeichert. Ein wesentlicher Unterschied in der Speicherung zum ersten Schritt ist die Differenz zwischen Name und Element. Im Gegensatz zum Element selbst, wo Name und Element gleich sind, wird beim Attribut unter Name der Attributname gespeichert und im Feld Element der Name des zugehörigen Elements. Außerdem gibt es beim Attribut keine Information über einen Zurück-Button, da dieser nicht benötigt wird. Das Ergebnis ist nun eine Sammlung von Datenobjekten, wo jedes Attribut und Element als eigener Datensatz gespeichert ist. Im vierten Schritt werden die eben gespeicherten Datensätze durchsucht, wobei alle der Ebene 0 und 1 für das Menü gesondert markiert werden (MenüItem-Boolean wird auf „true“ gesetzt). Im nächsten Schritt wird für jedes Datenobjekt das Elternelement gesucht. Das heißt, es wird jedes Datenobjekt ausgelesen und im verbleibenden Rest der Datensätze, jenes Objekt gesucht, das dem Elternelement entspricht. Zu guter letzt werden noch die Kinderelemente ermittelt.

Diese Datenobjekte werden dann dem Generator übergeben, welcher diese bei der Web-Interface Generierung in den Templates benötigt. Um nun die Daten, welche zuvor dem VelocityContext übergeben werden müssen, in das Template einfließen zu lassen, bedarf es eines einzelnen Methodenaufrufes:

```
// Template Objekt erzeugen
Template template = Velocity.getTemplate(<TEMPLATE-NAME>);
// Vereint das Template mit den Daten
template.merge(<VelocityContext>, <Writer>);
```

Nach erfolgreichem Verschmelzen der Daten mit dem Template werden nun entweder eine Archiv-Datei oder einzelne Dateien generiert, je nach Auswahl im Wizard.

5.2.4 Konfiguration

Die in diesem Kapitel bisher besprochenen Komponenten sind alle notwendig, um das Plug-in in Eclipse 3.1.0 lauffähig zu starten. Bevor dies aber geschehen kann, sind noch kleinere Einstellungen für das Plug-in notwendig.

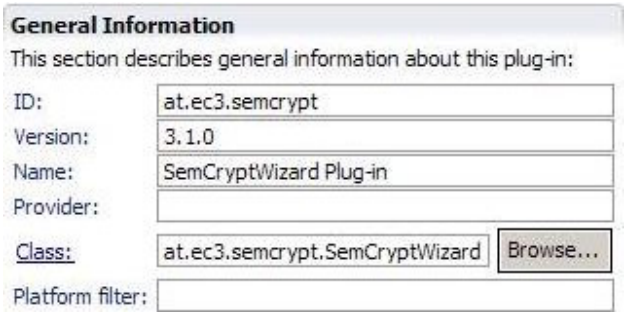

Abbildung 31: Allgemeine Informationen des Plug-ins

Abbildung 31 zeigt die allgemeinen Einstellungen für das Plug-in. Bis auf die Information „Class“, welche den genauen Package-Pfad der zu startenden Klasse angibt, sind alle anderen notwendigen Felder beliebig ausfüllbar. Es ist aber in Eclipse üblich, die „ID“ mit dem Package-Pfad zu benennen und die Versionsnummer mit der zu entwickelnden Eclipse-Version zu vergeben, also mit 3.1.0.

Des Weiteren müssen die für das Plug-in erforderlichen Plug-ins unter dem Punkt „Dependencies“ angegeben werden. Es sind dies folgende Plug-ins: org.eclipse.ui und org.eclipse.core.runtime. Diese Packages sind in der Eclipse 3.1.0 Installation bereits vorhanden und müssen somit nicht gesondert bezogen werden.

Das Setzen des Klassenpfades ist ebenso wichtig. Unter anderem muss das log4j, die Template-Engine, der Xalan-Transformer als auch das XMLBeans im Pfad mit seinen Quelldaten enthalten sein, um das Plug-in erfolgreich starten zu können.

Unter Punkt „Extensions“ wird einerseits die Kategorie angegeben (entspricht dem Namen, welcher beim Aufruf des Plug-ins im Menü vorzufinden ist) und andererseits jene Klassendatei, die das INewWizard Interface implementiert. Abbildung 32 beinhaltet die Ansicht des Wizards mit der Detailansicht. In dieser wird zusätzlich zur „Id“, dem „name“ und der „class“ Information mit dem Package-Pfad auch ein „icon“ Pfad abgefragt. Das Piktogramm (icon) wird im Menü beim Aufruf des Plug-ins angezeigt.

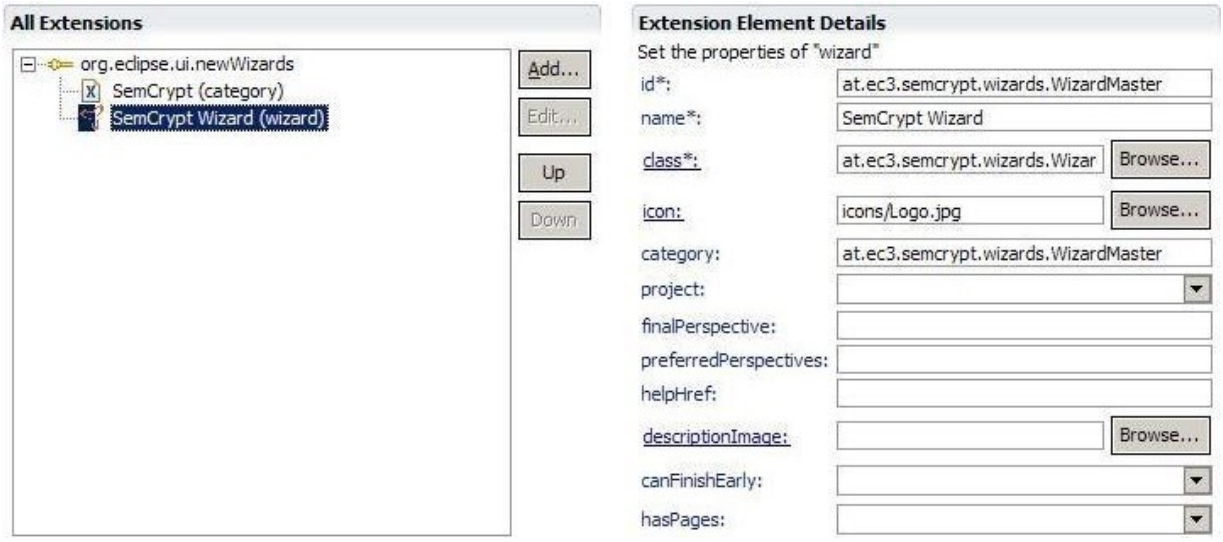

Abbildung 32: Plug-in Extension

Über die „Build Configuration“ werden alle Dateien ausgewählt, die in das Build aufgenommen werden sollen. Wenn man hier von einem Build spricht, so meint man das Verpacken in eine Archiv-Datei jener Dateien, die in der „Build Configuration“ definiert wurden. Dies beinhaltet in der Regel die kompilierten Klassendateien, deren zusätzliche Bibliotheksdateien und die Dateien, die für das Plug-in notwendig sind. Die Daten eben, die in den letzten Punkten besprochen wurden (siehe Abbildung 33).

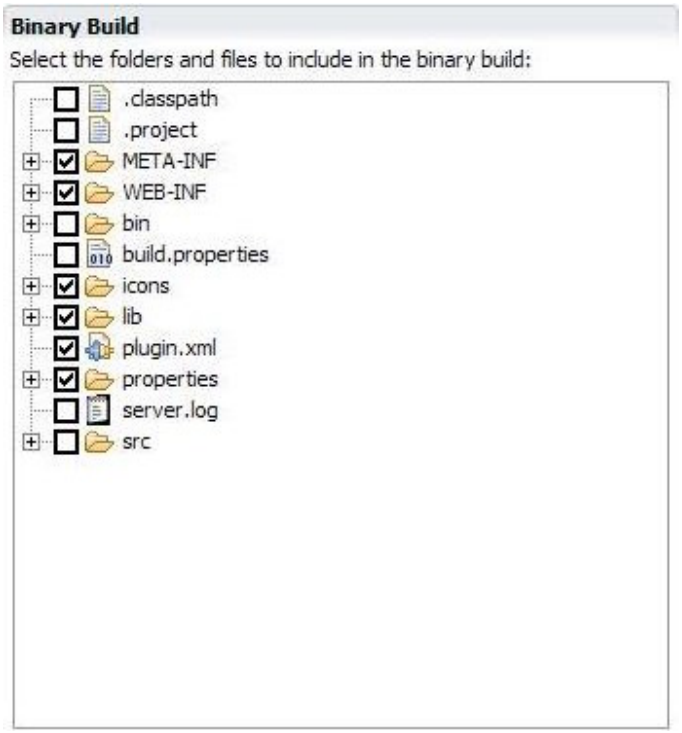

Abbildung 33: Auswahl der Dateien für den Build

Das wären jetzt alle notwendigen Einstellungen für das Plug-in. Alle diese Daten finden sich in drei verschiedenen Dateien wieder: MANIFEST.MF, plugin.xml und build.properties. Eclipse bietet die besprochenen Konfigurationen in einem GUI, wie die Abbildungen gezeigt haben, an, für geübte Anwender ist es aber möglich, die Einstellungen direkt in die eben genannten Dateien zu schreiben.

5.3 Servlets

Mit dem Login-Servlet beginnt der eigentliche Ablauf. Nach erfolgreicher Autorisierung wird der entsprechende Berechtigungsgrad in Form einer Rolle an das MainServlet geschickt. Das MainServlet ist für die Einbettung des XForms-Quellcodes, entsprechend der Rolle, in das Zieldokument verantwortlich. Der Converter nimmt die abgeschickten XML-Daten des MainServlets auf, ändert diese in XPath- bzw. XUpdate-Statements um und schickt diese weiter an die Autorisierungsstelle, bevor sie an eine Datenbank weitergeleitet werden. Man kann aber auch vom MainServlet direkt an eine Datenbank weiterleiten und den Converter umgehen. Die Verwendung ist optional, nur müsste dann aber die Anwendung mit den empfangenen XML-Daten auch umzugehen wissen. Abbildung 34 zeigt die Basis-Servletstruktur.

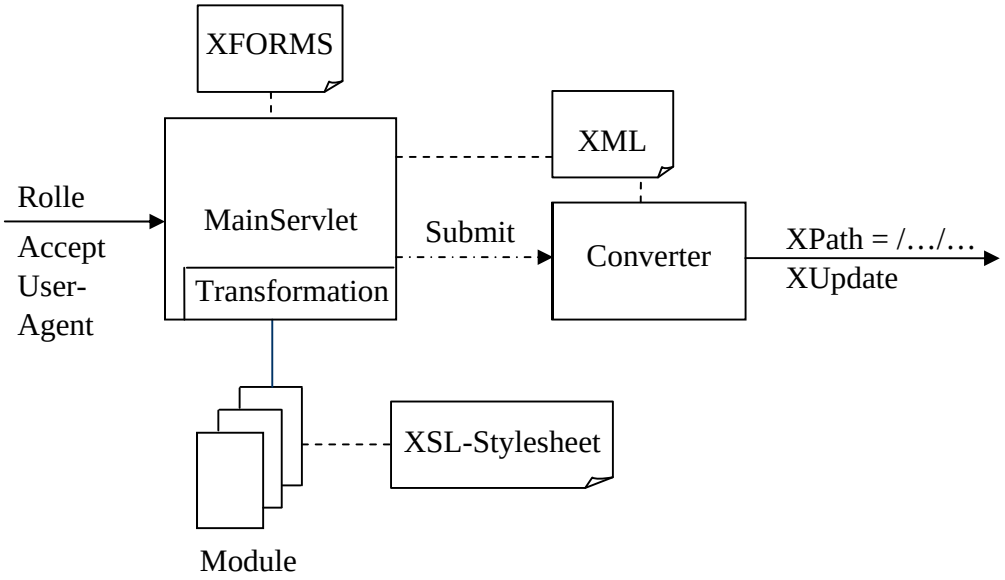

Abbildung 34: Servletarchitektur

5.3.1 MainServlet

Das generierte Web-Interface besteht letztendlich aus reinen XForms Anweisungen. Um nun das Web-Interface, z. B. in einem Browser, darstellen zu können, bedarf es noch Änderungen. Wie bereits in Kapitel 2.4.5 angesprochen, muss ein XForms Dokument in ein bestehendes eingebettet werden. Das heißt also, dass für das eben angesprochene Beispiel, der Quellcode des XForms-Web-Interfaces in ein HTML bzw. XHTML Dokument eingebettet werden muss. Das gleiche gilt genauso für ein Handy. Hier müsste es in ein WML Dokument eingebettet werden (Voraussetzung dabei ist, dass der zuständige Parser XForms unterstützt).

Hierbei nutzt man die XSL-Transformation aus, welche die notwendigen Einbettungen durchführt. Daher ist für jedes Zielgerät ein eigenes XSL-Skript notwendig. Das richtige XSL-Transformationsskript wird durch den Accept- und UserAgent-Header aus dem HTTPServletRequest bestimmt, indem beide Werte mit jedem Modul verglichen werden („accept“ Methode). Ein Auszug aus dem FormsPlayer-Modul:

```
// Bestimmung des XSL-Skripts
public boolean accept(String acceptHeader, String userAgentHeader) {
  if ((acceptHeader.indexOf("application/xhtml+xml") != -1 &&
       acceptHeader.indexOf("*/*") != -1 || acceptHeader.indexOf("*/*") != -1)
       && userAgentHeader.indexOf("formsPlayer") != -1) {
      return true;
  }
  return false;
}
```

Aber nicht nur das richtige XSL-Skript, sondern auch die zugehörigen Stylesheets und der Content Type werden aus dem Modul eruiert. Ein Modul ist eine Java-Klasse, welche neben der „accept“-Methode noch weitere Methoden besitzt und das Interface IModule.java implementiert:

```
public interface IModule {
  public String getName();
  public String getStyleSheet();
  public String getLoginStyleSheet();
  public String getContentType();
  public String getCSS();
  public String getLoginCSS();
  public void setCSS(String css);
  public void setLoginCSS(String css);
  public boolean accept(String acceptHeader, String userAgentHeader);
}
```

Außerdem muss noch jedes Modul zusätzlich im „modules.properties“ bekannt gegeben werden:

```
// Alle vorhandenen Module werden eingetragen
// Module name, XSL stylesheet, CSS file, XSL login stylesheet, CSS login file
module1=HTMLNovellModule,htmlnovellstyle,xforms,htmlloginnovellstyle,login
module3=MarkupModuleTest,html,xforms,htmllogin,login
module4=MarkupModuleWML
...
```

Hier wird neben dem Namen des Moduls (dieser ist notwendig, da das Modul über Java-Reflection aufgerufen wird) auch noch die Stylesheet Information für XSL und CSS eingetragen. Somit ist es möglich, bei Adaptierungen bzw. neuen Geräteunterstützungen ein neues Modul mit entsprechender Implementierung hinzuzufügen. Jedes unterstützte Gerät, wie Browser, Handy, etc., muss jeweils ein Modul besitzen, wodurch somit für jedes Gerät ein eigenes Stylesheet möglich ist.

Des Weiteren ist das MainServlet noch zuständig, dass die richtige Seite für einen Anwender aufgerufen wird. Gemeint ist hier, dass ein Anwender einer Benutzergruppe „Mitarbeiter“ nur jene Seite aufrufen kann, die seiner Berechtigung auch entspricht. Dies geschieht dadurch, dass dem Servlet die akzeptierte Rolle des Anwenders mitgeschickt wird, anhand welcher dann die richtige Seite geladen wird.

5.3.2 Converter

Wurde eine Datenbankabfrage vom Web-Interface durchgeführt (über Insert-, Append-, Delete-, Update- oder Query-Button), so schickt das MainServlet die Instanzdaten des XForms-Models

(XML-Daten) an den Converter. Die Instanzdaten beinhalten alle vom Anwender ausgewählten und eingegebenen Daten. Im Converter werden nun die XML-Daten mit dem DOM-Parser gelesen und in XPath- bzw. XUpdate-Anweisung zusammengebaut. Nachfolgend je ein Beispiel:

```
// XPath
?xpath=/SemCryptEmployeeInfo/Competencies/Competency
// XUpdate
<xupdate:modifications version="1.0"
                                xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:insert-before
                    select="/SemCryptEmployeeInfo/PersonInfo/@unitOfMeasure">
    <xupdate:attribute name="newAttribute">attributeValue</xupdate:attribute>
  </xupdate:insert-before>
</xupdate:modifications>
```

Wie hier ersichtlich ist, stellt die XUpdate Anweisung den wesentlich größeren Aufwand dar. Die Realisierung der XUpdate-Generierung erfolgt daher ebenfalls durch den Code Generator Velocity. Anschließend das Velocity-Template zur Generierung jeglicher XUpdate-Anweisungen:

```
<?xml version="1.0" encoding="utf-8"?>
<xupdate:modifications version="1.0"
      xmlns:xupdate="http://www.xmldb.org/xupdate">
  #foreach($xupdate in $xupdates)
    <xupdate:$xupdate.getXUpdate() select="<select>">
      #if($xupdate.getElemAttr().length() != 0)
        <xupdate:$xupdate.getElemAttr() name="<name>">
          $xupdate.getValue()
        </xupdate:$xupdate.getElemAttr()>
      #elseif($xupdate.getValue().length() != 0)
        $xupdate.getValue()
      #end
    </xupdate:$xupdate.getXUpdate()>
  #end
</xupdate:modifications>
```

Mit XForms alleine, wäre eine Erstellung einer XUpdate- bzw. XPath-Anweisung nicht möglich, da der Zusammenbau ziemlich kompliziert ist, was das Velocity-Template für die XUpdate-Anweisung ziemlich deutlich macht. Dieses eine Template erzeugt entweder eine Delete-, Insert-, Append- oder Update-Anweisung. Ist alles fertig zusammengesetzt, wird noch die Zieladresse

eingefügt und die nun fertige Anfrage, mit der „sendRedirect“-Methode der Klasse HTTPServletResponse, weitergeleitet.

5.3.3 Login

Dieses Servlet lädt entsprechend den empfangenen Header-Daten im HTTPServletRequest das zugehörige Modul mit den Stylesheet Daten. Anhand dieser Daten kann dann die Seite im jeweiligen Gerät dargestellt werden. Wie der Name schon sagt, wird mit diesem Servlet die Autorisierung vorgenommen. Der Anwender wird nach Name und Passwort gefragt, welche dann an den Autorisierungsserver (nicht Teil dieser Arbeit) geschickt und validiert werden. Stimmen Passwort und Anwendername mit einem der gefundenen Datensätzen überein, so wird an das MainServlet weitergeleitet, wobei die zugehörige Rolle des Anwenders ebenfalls mitgeschickt wird.

6 Ergebnisse der Arbeit

Nach den wichtigen Kapiteln Entwurf und Implementierung sollte ein guter Überblick über den User-Interface Generator geschaffen worden sein. In diesem Kapitel wird nun das Endprodukt vorgestellt. Außerdem wird auf Probleme eingegangen, die während der Entwicklung auftraten. Das soll dem Entwickler bei eventuell zukünftigen Erweiterungen bzw. Modifikationen behilflich sein.

6.1 Eclipse Plug-in

Wie bereits erwähnt, ist das Eclipse Plug-in kompatibel mit Eclipse 3.1.0. Installieren lässt sich das Plug-in wie andere Plug-ins auch, indem man die Plug-in Datei (.jar) in das „plugin“ Verzeichnis kopiert, danach kann man Eclipse starten. Um nun das Plug-in zu starten, muss man im Menü „File“ das Untermenü „Other“ anklicken. Danach erscheint das in Abbildung 35 ersichtliche Fenster, wo letztendlich der User-Interface Generator zu starten ist.

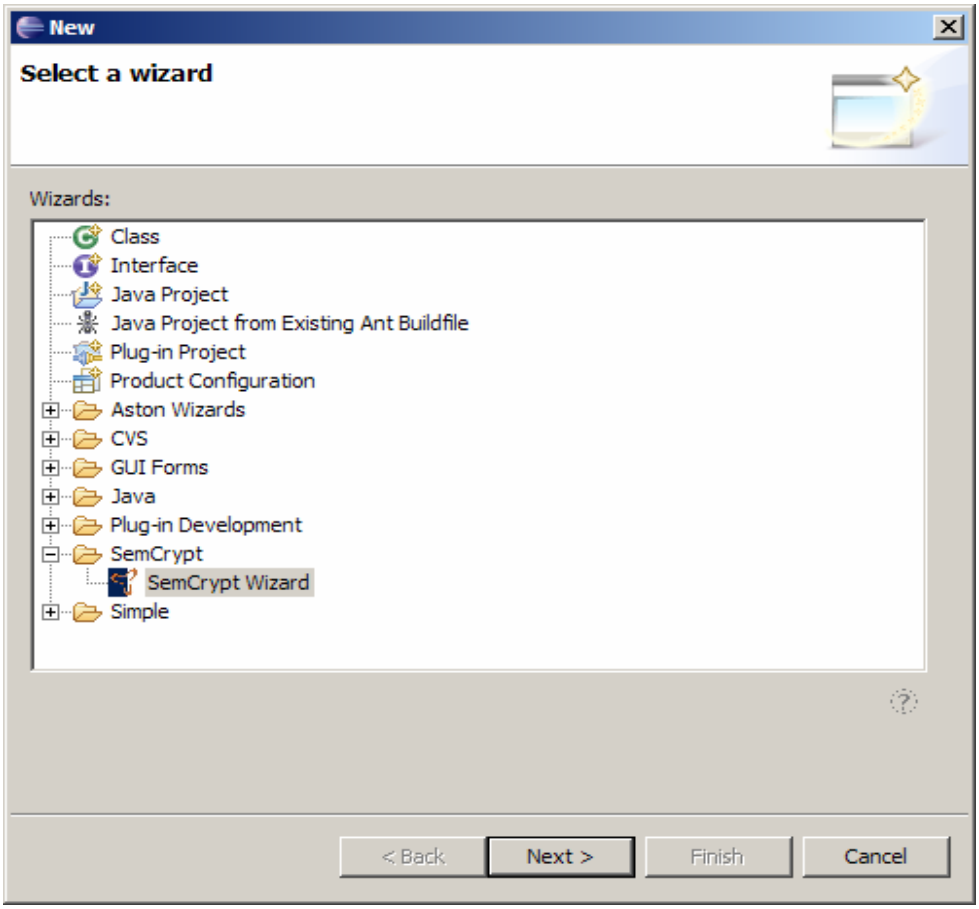

Abbildung 35: SemCrypt Plug-in starten

Als nächstes erscheint die WizardQuestionPage (siehe Abbildung 36), welche die Projektdaten beinhaltet. Sind diese ausgefüllt, erreicht man mit dem „Next“-Button die WizardTreePage, die in Abbildung 37 dargestellt ist.

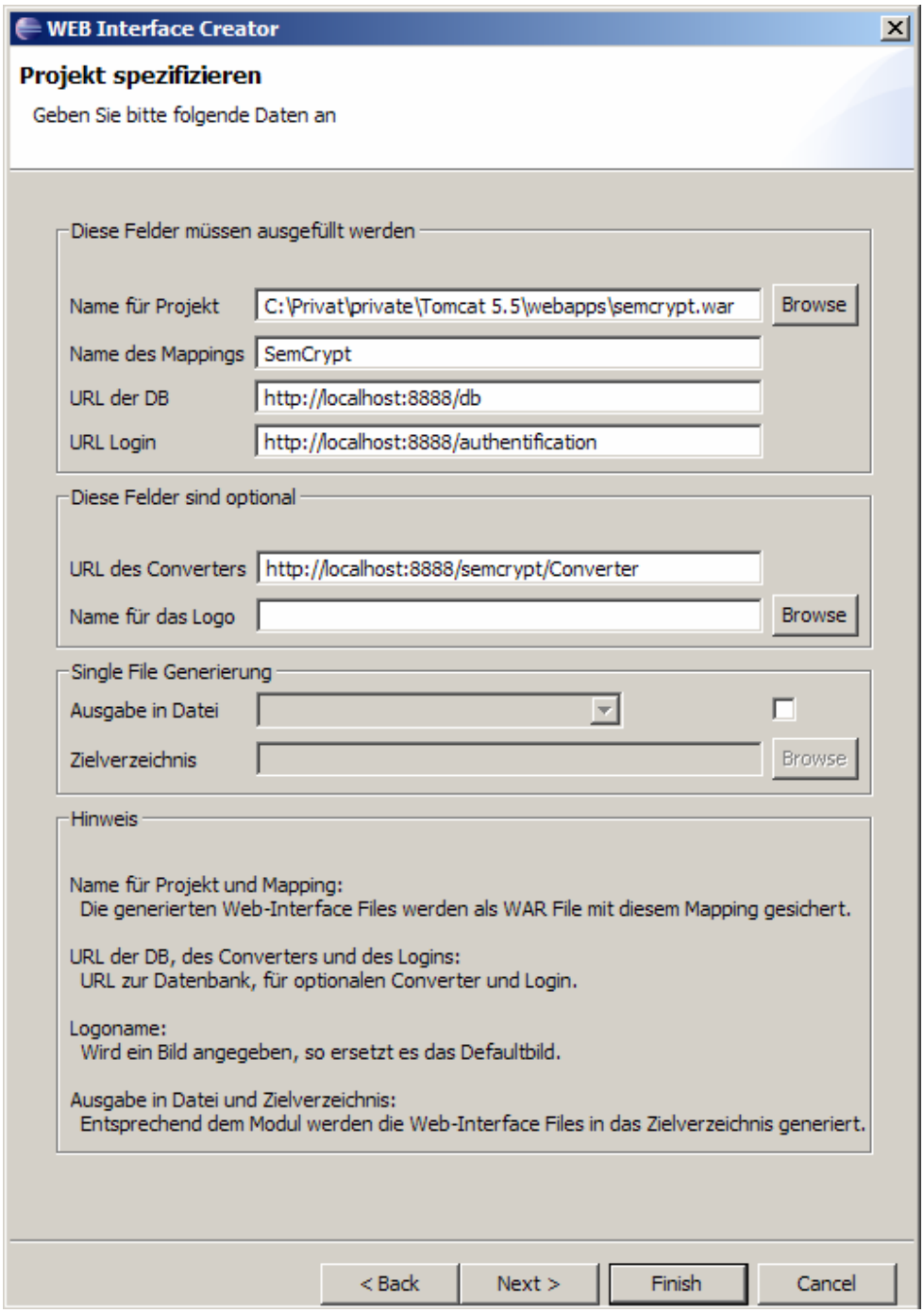

Abbildung 36: WizardQuestionPage

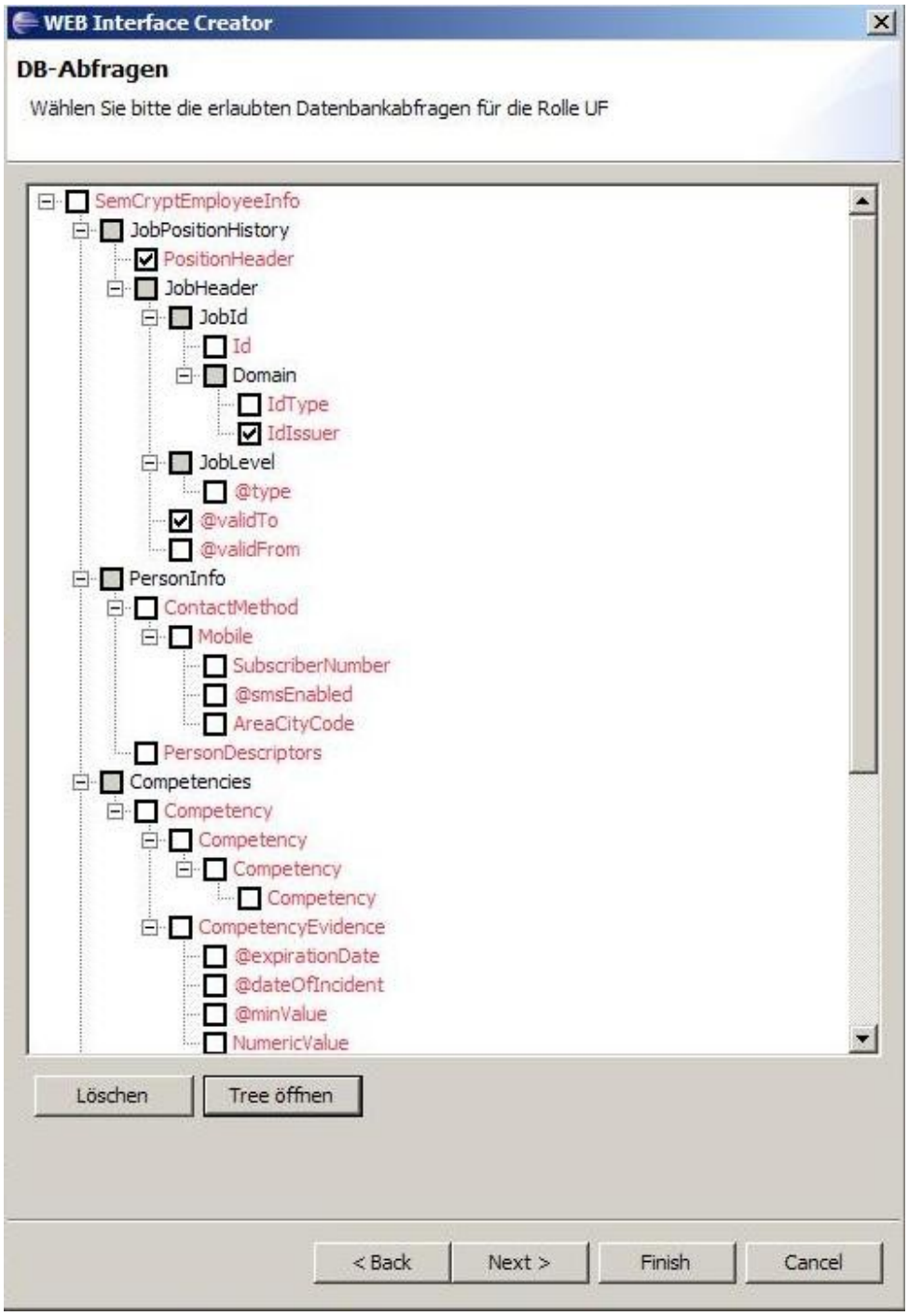

Abbildung 37: WizardTreePage

In eben genannter Abbildung werden lediglich die XPaths der Rolle UF (Untermehmensführung) gezeigt. Zu den verbleibenden Rollen gelangt man mit dem „Next“-Button. Zusätzliche Einschränkungen für ein Element bzw. Attribut sind durch einen Klick auf den entsprechenden Knoten möglich. Abbildung 38 zeigt das geöffnete Fenster nach Anklick des Attributs „dateOfIncident“.

Abbildung 38: Restriktionsfenster eines ausgewählten XPaths

Dieses Fenster enthält alle möglichen Zugriffsrechte und eine Liste, welche die Einschränkungen für den XPath darstellt. Durch das Schließen dieses Fensters wird das WizardTreePage-Fenster wieder aktiv.

Sind alle Daten zufrieden stellend eingegeben und ausgewählt worden, so aktiviert man den Generierungsvorgang mit dem „Finish“-Button. Je nach Art der Generierung (Einzeldaten- oder Archivdateigenerierung) ist dann das Web-Interface inklusive dem Login-GUI fertig erstellt.

6.2 Generiertes GUI

Abbildung 39 zeigt ein Beispiel des Web-Interfaces mit Falscheingabe im Feld „lastUsed“ und Abbildung 40 zeigt das Login Shell. Beide Beispiele wurden im Internet Explorer ausgeführt. Mit dem Login Shell soll sich ein SemCrypt Anwender im System anmelden. Bei erfolgreicher Authentifizierung wird dann auf das Web-Interface, mit dem entsprechenden Berechtigungsgrad, weitergeleitet. Hier kann dann der Anwender seine gewünschten Datenbankabfragen, etc. durchführen.

Für das generierte GUI sei noch anzumerken, dass eine Separierung der Operatoren, die in der Choicebox untergebracht sind, und des Eingabefeldes, nicht nur Vorteile, sondern auch Nachteile mit sich bringen. Vorteilhaft deswegen, weil somit eine Typenvalidierung auf den eingegebenen Wert durchgeführt werden kann, was sonst nicht möglich wäre, wenn man Wert und Operator im Eingabefeld angeben muss. Nachteilig vor allem dadurch, dass somit keine „UND“ / „ODER“ Verknüpfungen zugelassen werden können: Z. B. Value > 10 AND Value < 15. Diese Verknüpfungen sind bei herkömmlichen Datenbankabfragen kein Thema, ermöglichen aber komplexere Abfragen, auf welche aufgrund der Validierung verzichtet werden muss.

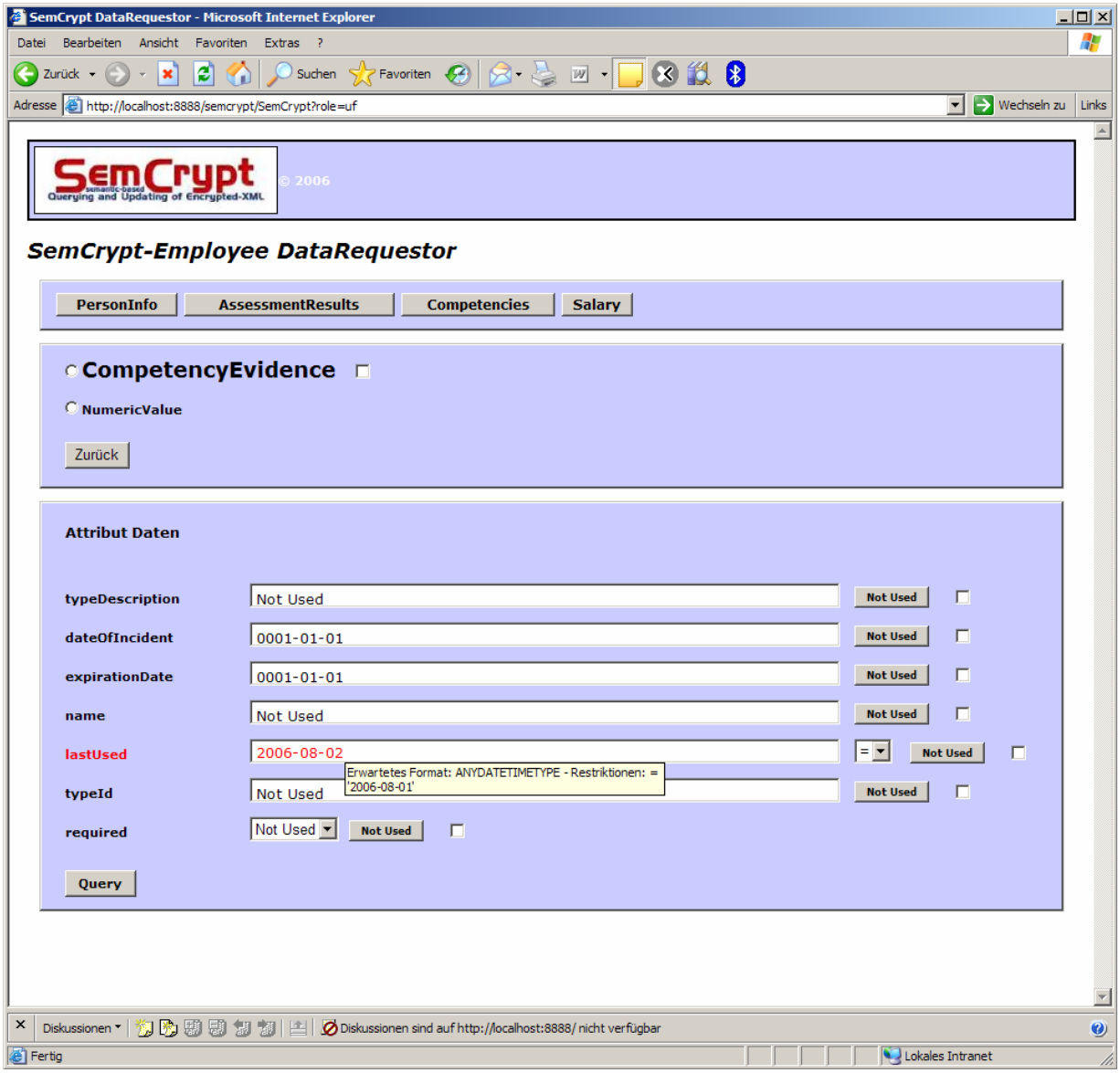

Abbildung 39: Erzeugtes Web-Interface dargestellt im IE (Novell Plug-in)

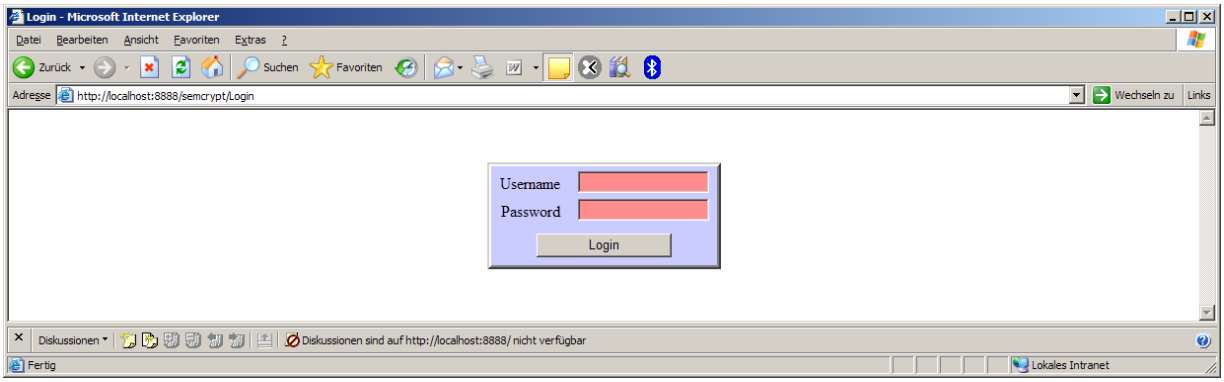

Abbildung 40: Erzeugtes Login GUI dargestellt im IE (Novell Plug-in)

Führt nun der Anwender eine Datenbankabfrage durch, so werden seine eingegebenen Daten (Instanzdaten) zum Converter geleitet. Dieser schickt dann die eigentliche Abfrage weiter. Ein vom Converter zusammengesetzter XPath, welcher zur Datenbank weitergeschickt wird, kann folgendermaßen aussehen:

```
http://<server>:<port>/db?xpath=/SemCryptEmployeeInfo/PersonInfo/ContactMethod
/Mobile/SubscriberNumber[/SemCryptEmployeeInfo/Competencies/Competency/Compete
ncyEvidence[./@dateOfIncident ge "2006-08-01" and ./@lastUsed eq "2006-08-01"
and ./@expirationDate eq "2006-08-31"] and .[. eq "436642042011"]]
```

Ein XUpdate-Statement so:

```
http://<server>:<port>/db?xupdate=<xupdate:modifications version="1.0"
xmlns:xupdate="http://www.xmldb.org/xupdate"><xupdate:insert-before
select="/SemCryptEmployeeInfo/PersonInfo/@unitOfMeasure"><xupdate:attribute
name="newAttribute">attributeValue</xupdate:attribute></xupdate:insert-
before></xupdate:modifications>
```

Das Eclipse Plug-in (Kapitel 6.1) und das Web-Interface (Kapitel 6.2) stellen den User-Interface Generator für SemCrypt Applikationen dar, wobei es sich hier um einen optionalen Teil des SemCrypt Projektes handelt.

6.3 Aufgetretene Probleme

Das Plug-in hat eigentlich keine nennenswerten Probleme verursacht, abgesehen von den Programmierproblemen, die hier aber nicht erwähnt werden. Die verwendete XForms Markup-Sprache war das Hauptproblem dieser Arbeit. Wäre XForms bereits besser verbreitet, so gäbe es wahrscheinlich weniger Probleme. So gibt es einige wenige XForms Plug-in Anbieter mit unterschiedlichen Entwicklungsständen und Implementierungsunterschieden von gleichen Komponenten. Entwicklungsrückstände lassen sich oft auf geringere Budgets zurückführen. Tragischer sind hier aber die Implementierungsdifferenzen gleicher Komponenten. Erwähnt sei hier ein Beispiel, in welchem die Form „select1“ mit dem Attribut „appearance = compact“ unterschiedlich dargestellt wird. Abbildung 41 zeigt dieses Beispiel, wobei linke Darstellung mit dem Novell Plug-in durchgeführt wurde und rechte Darstellung mit FormsPlayer.

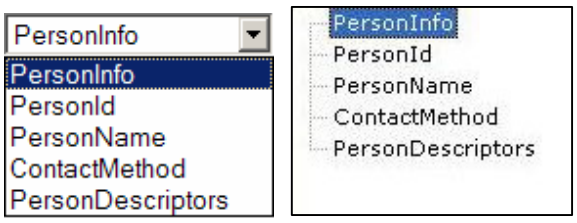

Abbildung 41: Unterschiedliche Darstellungsformen in Browsern

Außerdem sind noch einige Entwicklungsfehler in den Plug-ins enthalten, wie Abbildung 42 anhand der Formen „select“ und „select1“ für das Mozilla Plug-in Version 0.5 zeigt.

Section	Title	Status	Notes	Bugs
8.1.10	select	partial	@selection does not work.	282840;
8.1.11	select1	partial	There are some resize issues, select/deselect/valuechange firing in wrong order. Also missing some html:select capabilities.	282840; 303351; 303865; 333464; 333529;

Abbildung 42: Problemberichtsauszug des Mozilla XForms Projekts [57]

Die unterschiedlichen Darstellungen in den verschiedenen Browsern mit Cascading Stylesheets erleichtern die Aufgabe nicht. Weiters sind unterschiedliche Namespaces, wie bereits in Kapitel 4.5.5 beschrieben wurde, mögliche Fehlerquellen. Die angesprochenen Probleme finden sich auf ähnliche Weise bei Chiba als auch beim Orbeon Presentation Server wieder. In beiden Fällen ist es zurzeit nicht möglich, dass nach Selektierung eines Radio-Buttons der entsprechende XForms-Event ausgelöst wird, was eine weitere Navigation somit nicht zulässt.

Zirka jedes zweite bis dritte Monat kommt eine neue Release eines Plug-in-Updates heraus. Beim Chiba-Projekt ist seit Anfang des Jahres nicht viel passiert. Bei Orbeon ist die Projekt-Homepage aktuell gehalten, was die Entwicklung betrifft, kann man hier wenig sagen. Trotzdem geht die Entwicklung generell voran.

7 Zusammenfassung und Ausblick

Geräteunabhängige Web-Interfaces werden immer wichtiger. Das zeigt auf jeden Fall der steigende Anteil an mobilen Geräten in den letzten Jahren. Gleichzeitig ist hier ein großes Kundenpotenzial für das Internet zu erkennen. Mit dieser Trendentwicklung der letzten Jahre wurden viele Projekte ins Leben gerufen, welche sich die Entwicklung von geräteunabhängigen Interface-Sprachen zum Ziel setzten. XForms war eine davon und machte ursprünglich einen sehr guten Eindruck. Als Nachfolger von HTML angepriesen und bereits im Oktober 2003 als W3C Empfehlung [43] abgesegnet, begann es einen guten Start.

Mit den in Kapitel 6.3 beschriebenen Problemen wird die Euphorie aber ein wenig gebremst. Wie weit sich XAML von Microsoft mit der Einführung von Windows Vista [34] im nächsten Jahr behaupten wird, ist fraglich. Fest steht nur, dass Microsoft mit XAML einen eigenen Weg geht, welcher für die unabhängige Darstellung nicht förderlich sein wird. XAML wird nur, wie bei Microsoft nicht anders zu erwarten ist, auf Windows-Systemen lauffähig sein.

Eventuelle Alternativen generell zu Markup-Sprachen wären Frameworks wie JavaServer Faces [60] gewesen. Trotzdem sind Markup-Sprachen, allen voran XForms, vorzuziehen. Dies hat auch schon HTML gezeigt. Mit der Einführung von XHTML2.0, in welcher XForms standardmäßig inkludiert sein soll, wird sich aber schließlich weisen, ob XForms letztendlich die notwendige Akzeptanz finden wird oder nicht. Zurzeit aber sind Plug-ins für Browser notwendig, um XForms darstellen zu können. Andere Geräte wie Handys oder PDAs unterstützen XForms bislang nicht, da Plug-ins bzw. Erweiterungen für diese Geräte nicht existieren. Gerade im mobilen Sektor boomen Klingeltöne, Spiele, usw., welche über ein geringes Entgelt erhältlich sind. Und hier könnte ein Problemgrund liegen, dass erst Adaptierungen gemacht werden, wenn man sich Profit erwartet – also wenn XForms im Internet weitläufiger verbreitet ist. Ein anderer Grund könnte sein, dass man auf die Einführung von XHTML2.0 wartet und die vorherrschende Akzeptanz beobachtet. Dann aber ebenfalls von z. B. WML auf XHTML umsteigt und nicht bereits jetzt die Erweiterung für XForms auf WML durchführt, um so einen Implementierungsschritt zu sparen.

Zur Realisierung des User-Interface Generators seien hier noch zwei wichtige Punkte anzumerken, welcher einerseits die Verwendung des Converter-Servlets betrifft und andererseits den Einsatz von Velocity. Anfangs wurde nicht bedacht, dass es vielleicht unmöglich sein könnte mit XForms XPath- bzw. XUpdate-Anweisungen zusammenzustellen. XForms ist darauf ausgerichtet, dass es in Online-Shops, etc. Verwendung findet. Dort ist es üblich, dass eingegebene Daten in den Instanzdaten einfach weitergesendet werden. In dieser Arbeit war es aber wichtig, einzelne Attributsfelder wie auch Elementwerte zu ändern und anschließend in einem XPath bzw. XUpdate darzustellen. Diese Zusammenführungen sind mit XForms nicht mehr realisierbar und erforderten daher den Einsatz des Converter-Servlets. Man könnte aber diese Arbeit auch einer Datenbank überlassen, welche sich die XML-Daten selbst aufbereitet.

Der Einsatz von Velocity als Code Generator war ursprünglich nicht geplant gewesen, was sich aber nach kurzer Zeit bereits als ein kompliziertes Unterfangen herausstellte. Auch beim Zusammenbau der XUpdate-Anweisungen im Converter ist die Verwendung von Velocity angebracht. Durch den Einsatz eines Templates ist es auch noch nachträglich möglich Änderungen einfach durchzuführen.

Ist zukünftig eine Datenbankabfrage mit „UND“ / „ODER“ Verknüpfungen erwünscht, muss die XForms-Validierung herausgenommen werden und gänzlich am Server oder gleich in der Datenbank durchgeführt werden. Dies kann aber dazu führen, dass böswillige Eingaben zu unerlaubten Ausgaben aus der Datenbank führen.

Zusätzliche Erweiterungen im User-Interface Generator könnten in zukünftigen Arbeiten realisiert werden. Das wäre unter anderem ein Tool, welches die Erstellung eines Moduls für ein Endgerät ermöglicht. Dies müsste anhand des vorgegebenen Interfaces (IModule.java) eine Klasse, mit zugehörigem Accept- und UserAgent-Header entsprechend dem Endgerät, generieren.

Des Weiteren wäre ein Tool vorstellbar, welches bei der Erstellung eines XSL-Skripts für das neue Endgerät behilflich sein könnte. In wie weit das realistisch umsetzbar ist, müsste vorher analysiert werden, da jedes neue Gerät eine eigene neue Zielsprache besitzt, in welcher letztendlich der XForms-Quellcode eingebettet werden muss.

8 Referenzen

1. Integrated Development Environment – Glossar, Siemens AG Österreich, http://www.pse.siemens.at/apps/pseauftritt/ge/pseinternet.nsf/CD_Index?OpenFrameset&Bookmark&/0/PKAB153C0DB47CFDF7C12569EE003A4A3C

2. PDE Guide, Eclipse Foundation, 2005, http://www.eclipse.org/documentation

3. About Us, Eclipse Foundation, 2006, http://www.eclipse.org/org

4. Platform Plug-in Developer Guide, Eclipse Foundation, 2005, http://www.eclipse.org/documentation

5. Eclipse Download, Eclipse Foundation, http://www.eclipse.org/downloads

6. Eclipse Core 3.1.x, Eclipse Foundation, http://www.eclipse.org/jdt/core/r3.1/index.php#updates

7. Ullenboom C.: „Java ist auch eine Insel", Galileo Computing, 2004, ISBN 3-89842-526-6

8. Sun Developer Network, Sun Microsystems, http://java.sun.com

9. XSL Transformations (XSLT), Version 1.0, 1999, http://www.w3.org/TR/xslt

10. XML Path Language (XPath), Version 1.0, 1999, http://www.w3.org/TR/xpath

11. Markup Sprachen für User Interface Defintions, Cover Pages, 2005, http://xml.coverpages.org/userInterfaceXML.html

12. Goll J., Weiß C., Müller F.: „Java als erste Programmiersprache – Vom Einsteiger zum Profi", Teubner, 2001, ISBN 3-519-22642-1

13. HR-XML Consortium, Cover Pages, 2003, http://xml.coverpages.org/hr-xml.html

14. Holzner S.: „Eclipse", O'Reilly, 2004, ISBN 0-596-00641-1

15. Markup language, Wikipedia, 2006, http://en.wikipedia.org/wiki/Markup_language

16. Code Generation Network, http://www.codegeneration.net

17. Jostraca Code Generator, Jostraca, http://www.jostraca.org

18. GSLgen, iMatix, http://www.imatix.com/html/gslgen/index.htm

19. Velocity, Apache Jakarta Project, http://jakarta.apache.org/velocity

20. Nvelocity, http://nvelocity.sourceforge.net

21. CodeSmith, CodeSmith Tools, http://www.ericjsmith.net/codesmith

22. MetaL, http://www.meta-language.net

23. MetaStorage, http://www.meta-language.net/metastorage.html

24. Turbine Framework, Apache Jakarta Project, http://jakarta.apache.org/turbine

25. Torque, Apache DB Project, http://db.apache.org/torque

26. Herrington J.: „Code Generation in Action“, Manning, 2003, ISBN 1-930110-97-9

27. SGML, Overview of SGML, 1996, http://www.w3.org/MarkUp/SGML

28. XML, Introduction, http://www.w3.org/XML

29. Xu P.: „Vergleich von verschiedenen XML-GUI-Standards“, Bachelorarbeit, Universität Hannover, 2005, http://www.se.uni-hannover.de/documents/studthesis/BSc/Peng_Xu-XMLGUI_Vergleich.pdf

30. XUL Planet, http://www.xulplanet.com

31. Making Mozilla work for you, Creating Applications with Mozilla, http://books.mozdev.org/html/mozilla-chp-2-sect-3.html

32. Harmonia, http://www.harmonia.com

33. UIML3.1 Draft Specification, http://www.uiml.org/specs/uiml3/DraftSpec.htm

34. Windows Vista, Microsoft, http://www.microsoft.com/Windowsvista

35. Windows Presentation Foundation, Microsoft, http://msdn.microsoft.com/windowsvista/experience

36. WinFX, Wikipedia, 2006, http://en.wikipedia.org/wiki/WinFX

37. XIML WhitePaper, RedWhale inc., 2001, http://www.ximl.org/documents/XimlWhitePaper.pdf

38. XIML HomePage, RedWhale Inc., http://www.ximl.org

39. Simple Dictionary Example, XIML StarterKit Version 1.0, http://www.ximl.org/download/step1.asp

40. Raman T. V.: „XForms - XML Powered Web Forms”, Addison-Wesley, 2004,

ISBN 0-321-15499-1

41. Dubinko M.: „XForms Essentials“, O’Reilly, 2003, ISBN 0-596-00369-2

42. XHTML 2.0, Working Draft, 27. Mai 2005, http://www.w3.org/TR/xhtml2

43. XForms 1.0, W3C Empfehlung 14. Oktober 2003, http://www.w3.org/TR/2003/REC-xforms-20031014

44. XForms 1.0 (Second Edition), W3C Empfehlung 14 März 2006, http://www.w3.org/TR/xforms

45. XML Events, W3C Empfehlung 14. Oktober 2003, http://www.w3.org/TR/xml-events

46. Beutenmüller C., C. Lehmann C.: „Einführung in modellbasierte XML – Sprachen für Benutzerschnittstellen“, 2005, http://ebus.informatik.uni-leipzig.de/www/media/lehre/uiseminar05/uiseminar05-beutenmueller-lehmann-folien.pdf

47. Beutenmüller C.: „Deklarative XML-Sprachen für Benutzerschnittstellen“, Seminararbeit, Universität Leipzig, 2005 http://ebus.informatik.uni-leipzig.de/www/media/lehre/uiseminar05/ausarbeitung-beutenmueller.pdf

48. Chiba, Open Source Java Implementierung des W3C XForms Standards, http://chiba.sourceforge.net

49. Orbeon, Open Source Orbeon PresentationServer für Standard XForms, http://www.orbeon.com

50. XUpdate Working Draft, 14. September 2000, http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html

51. Logging Services - Log4j, Apache Software Foundation, http://logging.apache.org/log4j/docs/index.html

52. XMLBeans, Apache XML Projekt, http://xmlbeans.apache.org

53. Perkles S.: „Server – Side XForms Processing – A Browser independent implementation of the next Generation Web Forms”, Diplomarbeit, TU-Wien, 2003

54. Bruchez E.: „Are Server-Side XForms Engines the Future of XForms”, Electronical paper, XTech, 2005, http://www.idealliance.org/proceedings/xtech05/papers/03-08-03

55. XHTML 2.0, W3C Working Draft 26 July 2006, http://www.w3.org/TR/xhtml2

56. XForms Novell Plug-in, Novell, http://developer.novell.com/xforms

57. XForms Mozilla Plug-in, Mozilla.org, http://www.mozilla.org/projects/xforms

58. XForms Formsplayer Plug-in, FormsPlayer, http://www.formsplayer.com

59. Karlinger M., Grün K.: „Recommended XPath and XUpdate Support", 2005

60. JavaServer Faces, Sun Microsystems, http://java.sun.com/javaee/javaserverfaces

61. Schrefl M., Grün K., Dorn J.: „SemCrypt – Ensuring Privacy of Electronic Documents Through Semantic-Based Encrypted Query Processing"

62. Weber M., Hrastnik P.: „WP1 – HR-Szenario", 2005

63. Schreiner W.: „WP2 – Architektur", 2005

64. Berkeley DB Developer Zone, Oracle, http://dev.sleepycat.com

65. Xindice, Apache XML Project, http://xml.apache.org/xindice

66. Human Resources, Wikipedia, 2006, http://de.wikipedia.org/wiki/Human_Resources

67. Dorn J., Schreiner W.: Security and Privacy Management in Virtual Enterprises, 2006

9 Anhang

9.1 XAccess-Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://semcrypt.ec3.at/xml/access/types"
        xmlns:tns="http://semcrypt.ec3.at/xml/access/types">
  <element name="XAccess" type="tns:XAccessType"/>
  <element name="SecurityRoles" type="tns:SecurityRolesType"/>
  <element name="SecurityRole" type="tns:SecurityRoleType"/>
  <element name="SecurityRights" type="tns:SecurityRightsType"/>
  <element name="SecurityRight" type="tns:SecurityRightType"/>
  <element name="SecurityPaths" type="tns:SecurityPathsType"/>
  <element name="SecurityPath" type="tns:SecurityPathType"/>
  <element name="Permissions" type="tns:PermissionsType"/>
  <element name="Permission" type="tns:PermissionType"/>
  <element name="Restrictions" type="tns:RestrictionsType"/>
  <element name="RecursionDepth" type="tns:RecursionDepthType"/>
  <element name="Type" type="tns:TypeType"/>
  <element name="PathValue" type="tns:PathValueType"/>
  <element name="Description" type="tns:DescriptionType"/>
  <element name="RoleName" type="tns:RoleNameType"/>
  <element name="RightName" type="tns:RightNameType"/>
  <element name="TextContent" type="tns:TextContentType"/>
  <element name="Attribute" type="tns:AttributeType"/>
  <complexType name="XAccessType">
    <sequence>
      <element ref="tns:SecurityRoles" maxOccurs="1" minOccurs="1"/>
      <element ref="tns:SecurityPaths" maxOccurs="1" minOccurs="1"/>
      <element ref="tns:SecurityRights" maxOccurs="1" minOccurs="1"/>
      <choice maxOccurs="1" minOccurs="1">
        <element ref="tns:Permissions" maxOccurs="1" minOccurs="1"/>
        <element ref="tns:Denials" maxOccurs="1" minOccurs="1"/>
      </choice>
    </sequence>
  </complexType>
```

```
<complexType name="SecurityRolesType">
  <sequence>
    <element ref="tns:SecurityRole" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<complexType name="SecurityRoleType">
  <sequence>
    <element ref="tns:Description" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:RoleName" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:BaseRole" maxOccurs="unbounded" minOccurs="0"/>
    <element ref="tns:Alias" maxOccurs="unbounded" minOccurs="0"/>
  </sequence>
</complexType>
<complexType name="SecurityRightsType">
  <sequence>
    <element ref="tns:RightsHandler"/>
    <element ref="tns:SecurityRight" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<complexType name="SecurityRightType">
  <sequence>
    <element ref="tns:Description" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:RightName" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:BaseSequence" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<simpleType name="TextContentType">
  <restriction base="boolean"/>
</simpleType>
<complexType name="SecurityPathsType">
  <sequence>
    <element ref="tns:Namespaces" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:SecurityPath" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
```

```
<complexType name="SecurityPathType">
  <sequence>
    <element ref="tns:Description" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:PathValue" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:Type" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:Mixed" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Prefix" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Recursive" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<simpleType name="RoleNameType">
  <restriction base="string"/>
</simpleType>
<simpleType name="RightNameType">
  <restriction base="string"/>
</simpleType>
<complexType name="RestrictionsType">
  <sequence>
    <element ref="tns:TextContent" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:RecursionDepth" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Path" maxOccurs="unbounded" minOccurs="0"/>
    <element ref="tns:Extension" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<simpleType name="RecursionDepthType">
  <restriction base="nonNegativeInteger"/>
</simpleType>
<complexType name="PermissionsType">
  <sequence>
    <element ref="tns:Permission" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<complexType name="PermissionType">
  <sequence>
    <element ref="tns:PathValue" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:RoleName" maxOccurs="unbounded" minOccurs="1"/>
```

```
    <element ref="tns:RightName" maxOccurs="unbounded" minOccurs="1"/>
    <element ref="tns:RequiresRoles" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Restrictions" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<simpleType name="PathValueType">
  <restriction base="string"/>
</simpleType>
<simpleType name="TypeType">
  <restriction base="string"/>
</simpleType>
<simpleType name="DescriptionType">
  <restriction base="string"/>
</simpleType>
<complexType name="AttributeType">
  <sequence>
    <element ref="tns:Name" maxOccurs="1" minOccurs="1"/>
    <element ref="tns:Operation" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Value" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<element name="Name" type="tns:NameType"/>
<element name="Value" type="tns:ValueType"/>
<simpleType name="NameType">
  <restriction base="string"/>
</simpleType>
<simpleType name="ValueType">
  <restriction base="string"/>
</simpleType>
<element name="Path" type="tns:PathType"/>
<element name="Operation" type="tns:OperationType"/>
<simpleType name="OperationType">
  <restriction base="string"/>
</simpleType>
<element name="Mixed" type="tns:MixedType"/>
<simpleType name="MixedType">
```

```
  <restriction base="boolean"/>
</simpleType>
<element name="PathAttribute" type="tns:PathAttributeType"/>
<complexType name="PathAttributeType">
  <sequence>
    <element ref="tns:Name"/>
    <element ref="tns:Type"/>
  </sequence>
</complexType>
<complexType name="PathType">
  <sequence>
    <element ref="tns:Operation"/>
    <element ref="tns:Value"/>
  </sequence>
</complexType>
<element name="RequiresRoles" type="tns:RequiresRolesType"/>
<complexType name="RequiresRolesType">
  <sequence>
    <element ref="tns:RoleName" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<element name="BaseRole" type="tns:RoleNameType"/>
<element name="Extension" type="tns:ExtensionType"/>
<complexType name="ExtensionType">
  <complexContent>
    <extension base="anyType"></extension>
  </complexContent>
</complexType>
<element name="Namespace">
  <complexType>
    <sequence>
      <element ref="tns:URI"/>
      <element ref="tns:Prefix"/>
    </sequence>
  </complexType>
</element>
```

```
<simpleType name="NamespaceType">
  <restriction base="string"/>
</simpleType>
<element name="Namespaces" type="tns:NamespacesType"/>
<complexType name="NamespacesType">
  <sequence>
    <element ref="tns:Namespace" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<element name="Prefix" type="string"/>
<element name="URI" type="string"/>
<simpleType name="PrefixType">
  <restriction base="string"/>
</simpleType>
<simpleType name="URIType">
  <restriction base="anyURI"/>
</simpleType>
<element name="Denials" type="tns:DenialsType"/>
<element name="Denial" type="tns:DenialType"/>
<complexType name="DenialsType">
  <sequence>
    <element ref="tns:Denial" maxOccurs="unbounded" minOccurs="1"/>
  </sequence>
</complexType>
<complexType name="DenialType">
  <sequence>
    <element ref="tns:PathValue"/>
    <element ref="tns:RoleName" maxOccurs="unbounded" minOccurs="1"/>
    <element ref="tns:RightName" maxOccurs="unbounded" minOccurs="1"/>
    <element ref="tns:RequiresRoles" maxOccurs="1" minOccurs="0"/>
    <element ref="tns:Restrictions" maxOccurs="1" minOccurs="0"/>
  </sequence>
</complexType>
<element name="Recursive" type="tns:RecursiveType"/>
<simpleType name="RecursiveType">
  <restriction base="boolean"/>
```

```
  </simpleType>
  <element name="Alias" type="tns:AliasType"/>
  <simpleType name="AliasType">
    <restriction base="string"/>
  </simpleType>
  <element name="RightsHandler" type="tns:RightsHandlerType"/>
  <simpleType name="RightsHandlerType">
    <restriction base="string"/>
  </simpleType>
  <element name="BaseSequence" type="tns:BaseSequenceType"/>
  <complexType name="BaseSequenceType">
    <sequence>
      <element ref="tns:RightName" maxOccurs="unbounded" minOccurs="1"/>
    </sequence>
  </complexType>
</schema>
```

9.2 XPath.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<XAccess xmlns="http://semcrypt.ec3.at/xml/access/types"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://semcrypt.ec3.at/xml/access/types
         xaccess.xsd">
  <SecurityRoles>
    <SecurityRole>
      <Description>Unternehmensfuehrung</Description>
      <RoleName>UF</RoleName>
    </SecurityRole>
    <SecurityRole>
      <Description>Gruppenleiter</Description>
      <RoleName>GL</RoleName>
    </SecurityRole>
    <SecurityRole>
      <Description>Projektleiter</Description>
      <RoleName>PL</RoleName>
    </SecurityRole>
    <SecurityRole>
```

```
    <Description>Human Resource Management</Description>
    <RoleName>HR</RoleName>
  </SecurityRole>
  <SecurityRole>
    <Description>Bestimmte Gruppe von Mitarbeitern</Description>
    <RoleName>MA1</RoleName>
  </SecurityRole>
  <SecurityRole>
    <Description>Bestimmte Gruppe von Mitarbeitern</Description>
    <RoleName>MA2</RoleName>
  </SecurityRole>
</SecurityRoles>
<SecurityPaths>
  <SecurityPath>
    <Description>SemCryptEmployeeInfo</Description>
    <PathValue>/SemCryptEmployeeInfo</PathValue>
    <Type>anonymous</Type>
  </SecurityPath>
  <SecurityPath>
    <Description>Competencies</Description>
    <PathValue>/SemCryptEmployeeInfo/Competencies</PathValue>
    <Type>SemCrypt_CompetenciesType</Type>
  </SecurityPath>
  <SecurityPath>
    <PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
    <Type>anonymous</Type>
  </SecurityPath>
  <SecurityPath>
    <PathValue>/SemCryptEmployeeInfo/Competencies
                                          /Competency/@name</PathValue>
    <Type>string</Type>
  </SecurityPath>
  <SecurityPath>
    <PathValue>/SemCryptEmployeeInfo/Competencies
                                    /Competency/@description</PathValue>
    <Type>string</Type>
```

```
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
                                      /Competency/@required</PathValue>
  <Type>boolean</Type>
</SecurityPath>
<SecurityPath>
  <Description>Competency</Description>
  <PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
                             /Competency/CompetencyEvidence</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
                  /Competency/CompetencyEvidence/@minValue</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
            /Competency/CompetencyEvidence/@dateOfIncident</PathValue>
  <Type>AnyDateTimeType</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
            /Competency/CompetencyEvidence/@expirationDate</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies/Competency
                                /CompetencyEvidence/@typeId</PathValue>
  <Type>string</Type>
</SecurityPath>
```

```
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
             /Competency/CompetencyEvidence/@typeDescription</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>CompetencyEvidence</Description>
  <PathValue>/SemCryptEmployeeInfo/Competencies
                    /Competency/CompetencyEvidence/@lastUsed</PathValue>
  <Type>AnyDateTime</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
       /Competency/CompetencyEvidence/NumericValue/@minValue</PathValue>
  <Type>double</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
       /Competency/CompetencyEvidence/NumericValue/@maxValue</PathValue>
  <Type>double</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/Competencies
    /Competency/CompetencyEvidence/NumericValue/@description</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
<Description>NumericValue</Description>
  <PathValue>/SemCryptEmployeeInfo/Competencies
                 /Competency/CompetencyEvidence/NumericValue</PathValue>
  <Type>double</Type>
</SecurityPath>
<SecurityPath>
  <Description>JobPositionHistory</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory</PathValue>
  <Type>SemCrypt_JobPositionHistoryType</Type>
```

```
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                /JobHeader/@validFrom</PathValue>
  <Type>AnyDateTimeType</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                  /JobHeader/@validTo</PathValue>
  <Type>AnyDateTimeType</Type>
</SecurityPath>
<SecurityPath>
  <Description>JobHeader</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                           /JobHeader</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                             /JobHeader/JobLevel/@type</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>JobLevel</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                  /JobHeader/JobLevel</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>Code</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                              /JobHeader/JobLevel/Code</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>JobId</Description>
```

```
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                    /JobHeader/JobId</PathValue>
  <Type>JobIdentifierType</Type>
</SecurityPath>
<SecurityPath>
  <Description>Id</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                /JobHeader/JobId/Id</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>Domain</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                            /JobHeader/JobId/Domain</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>IdIssuer</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                   /JobHeader/JobId/Domain/IdIssuer</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>IdType</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                     /JobHeader/JobId/Domain/IdType</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                         /PositionHeader/@validFrom</PathValue>
  <Type>AnyDateTimeType</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                             /PositionHeader/@validTo</PathValue>
```

```
  <Type>AnyDateTimeType</Type>
</SecurityPath>
<SecurityPath>
  <Description>PositionHeader</Description>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                    /PositionHeader</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonInfo</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo</PathValue>
  <Type>SemCrypt_PersonInfoType</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonId</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo/PersonId</PathValue>
  <Type>EntityIdType</Type>
</SecurityPath>
<SecurityPath>
  <Description>ContactMethod</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo/ContactMethod</PathValue>
  <Type>ContactMethodType</Type>
</SecurityPath>
<SecurityPath>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                    /ContactMethod/Mobile/@smsEnabled</PathValue>
  <Type>boolean</Type>
</SecurityPath>
<SecurityPath>
  <Description>Mobile</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                                 /ContactMethod/Mobile</PathValue>
  <Type>MobileTelcomNumberType</Type>
</SecurityPath>
<SecurityPath>
  <Description>SubscriberNumber</Description>
```

```
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                  /ContactMethod/Mobile/SubscriberNumber</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>AreaCityCode</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                      /ContactMethod/Mobile/AreaCityCode</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonName</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo/PersonName</PathValue>
  <Type>PersonNameType</Type>
</SecurityPath>
<SecurityPath>
  <Description>FormattedName</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                                /PersonName/FormattedName</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonDescriptors</Description>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                                       /PersonDescriptors</PathValue>
  <Type>PersonDescriptorsType</Type>
</SecurityPath>
<SecurityPath>
  <Description>Salary</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary</PathValue>
  <Type>SemCrypt_SalaryType</Type>
</SecurityPath>
<SecurityPath>
  <Description>PayrollInstructions</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary/PayrollInstructions</PathValue>
  <Type>anonymous</Type>
```

```
</SecurityPath>
<SecurityPath>
  <Description>PayrollEmployer</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary
                   /PayrollInstructions/PayrollEmployer</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>EmployerName</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary
      /PayrollInstructions/PayrollEmployer/EmployerName</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonInstruction</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary
                /PayrollInstructions/PersonInstruction</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>PayrollPerson</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary
   /PayrollInstructions/PersonInstruction/PayrollPerson</PathValue>
  <Type>anonymous</Type>
</SecurityPath>
<SecurityPath>
  <Description>PersonName</Description>
  <PathValue>/SemCryptEmployeeInfo/Salary/PayrollInstructions
            /PersonInstruction/PayrollPerson/PersonName</PathValue>
  <Type>string</Type>
</SecurityPath>
<SecurityPath>
  <Description>AssessmentResults</Description>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults</PathValue>
  <Type>SemCrypt_AssessmentResultsType</Type>
</SecurityPath>
```

```
  <SecurityPath>
    <Description>ClientId</Description>
    <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                            /AssessmentResult/ClientId</PathValue>
    <Type>EntityIdType</Type>
  </SecurityPath>
  <SecurityPath>
    <Description>ProviderId</Description>
    <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                          /AssessmentResult/ProviderId</PathValue>
    <Type>EntityIdType</Type>
  </SecurityPath>
  <SecurityPath>
    <Description>Results</Description>
    <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                            /AssessmentResult/Results</PathValue>
    <Type>anonymous</Type>
  </SecurityPath>
  <SecurityPath>
    <Description>OverallScore</Description>
    <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                /AssessmentResult/Results/OverallScore</PathValue>
    <Type>AssessmentScoreType</Type>
  </SecurityPath>
  <SecurityPath>
    <Description>ScoreDetail</Description>
    <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                 /AssessmentResult/Results/ScoreDetail</PathValue>
    <Type>AssessmentScoreType</Type>
  </SecurityPath>
</SecurityPaths>
<SecurityRights>
  <RightsHandler/>
  <SecurityRight>
    <Description>allows user to enhance information</Description>
    <RightName>insert</RightName>
```

```
  </SecurityRight>
  <SecurityRight>
    <Description>enables user to retrieve existing information</Description>
    <RightName>read</RightName>
  </SecurityRight>
  <SecurityRight>
    <Description>permits user to alter existing information</Description>
    <RightName>update</RightName>
  </SecurityRight>
  <SecurityRight>
    <Description>allows user to remove existing information</Description>
    <RightName>delete</RightName>
  </SecurityRight>
  <SecurityRight>
    <Description>allows user to append information</Description>
    <RightName>append</RightName>
  </SecurityRight>
</SecurityRights>
<Permissions>
  <Permission>
    <PathValue>/SemCryptEmployeeInfo/PersonInfo
                          /PersonName/FormattedName</PathValue>
    <RoleName>GL</RoleName>
    <RightName>read</RightName>
    <RightName>update</RightName>
    <RightName>delete</RightName>
  </Permission>
  <Permission>
    <PathValue>/SemCryptEmployeeInfo/PersonInfo
                          /PersonName/FormattedName</PathValue>
    <RoleName>PL</RoleName>
    <RightName>read</RightName>
    <RightName>insert</RightName>
  </Permission>
  <Permission>
    <PathValue>/SemCryptEmployeeInfo/Competencies
```

```
                    /Competency/CompetencyEvidence</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Competencies
           /Competency/CompetencyEvidence/@minValue</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <Restrictions>
    <Path>
      <Operation>eq</Operation>
      <Value>2006-10-10</Value>
    </Path>
  </Restrictions>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Competencies
     /Competency/CompetencyEvidence/@dateOfIncident</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>update</RightName>
  <Restrictions>
    <Path>
      <Operation>ne</Operation>
      <Value>2099-10-10</Value>
    </Path>
  </Restrictions>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Competencies
     /Competency/CompetencyEvidence/@expirationDate</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>delete</RightName>
</Permission>
```

```
<Permission>
  <PathValue>/SemCryptEmployeeInfo</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo
                   /PersonInfo/PersonDescriptors</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Competencies
      /Competency/CompetencyEvidence/NumericValue</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo/ContactMethod</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
                            /ContactMethod/Mobile</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
               /ContactMethod/Mobile/@smsEnabled</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>insert</RightName>
</Permission>
<Permission>
```

```
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
           /ContactMethod/Mobile/SubscriberNumber</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/PersonInfo
              /ContactMethod/Mobile/AreaCityCode</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>append</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                           /JobHeader/@validFrom</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                             /JobHeader/@validTo</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                             /JobHeader/JobId/Id</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                /JobHeader/JobId/Domain/IdIssuer</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
```

```
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                   /JobHeader/JobId/Domain/IdType</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                         /JobHeader/JobLevel/@type</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults</PathValue>
  <RoleName>UF</RoleName>
  <RightName>delete</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                          /AssessmentResult/ClientId</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                        /AssessmentResult/ProviderId</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults
                                   /AssessmentResult</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>delete</RightName>
</Permission>
```

```
<Permission>
  <PathValue>/SemCryptEmployeeInfo/AssessmentResults
             /AssessmentResult/Results/ScoreDetail</PathValue>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>update</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/JobPositionHistory
                                  /PositionHeader</PathValue>
  <RoleName>MA1</RoleName>
  <RoleName>MA2</RoleName>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>update</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Salary/PayrollInstructions</PathValue>
  <RoleName>MA1</RoleName>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
  <RightName>append</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Salary
              /PayrollInstructions/PayrollEmployer</PathValue>
  <RoleName>MA1</RoleName>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
</Permission>
<Permission>
  <PathValue>/SemCryptEmployeeInfo/Salary
              /PayrollInstructions/PayrollEmployer</PathValue>
  <RoleName>MA1</RoleName>
  <RoleName>UF</RoleName>
  <RightName>read</RightName>
```

```
    </Permission>
    <Permission>
      <PathValue>/SemCryptEmployeeInfo/Competencies/Competency</PathValue>
      <RoleName>UF</RoleName>
      <RightName>read</RightName>
      <Restrictions>
        <RecursionDepth>3</RecursionDepth>
      </Restrictions>
    </Permission>
    <Permission>
      <PathValue>/SemCryptEmployeeInfo/Salary/PayrollInstructions
                  /PersonInstruction/PayrollPerson/PersonName</PathValue>
      <RoleName>UF</RoleName>
      <RightName>read</RightName>
    </Permission>
    <Permission>
      <PathValue>/SemCryptEmployeeInfo/Salary/PayrollInstructions
                               /PayrollEmployer/EmployerName</PathValue>
      <RoleName>MA1</RoleName>
      <RightName>read</RightName>
      <Restrictions>
        <Path>
          <Operation>eq</Operation>
          <Value>type</Value>
        </Path>
      </Restrictions>
    </Permission>
    <Permission>
      <PathValue>/SemCryptEmployeeInfo/Competencies
                        /Competency/CompetencyEvidence</PathValue>
      <RoleName>MA1</RoleName>
      <RightName>read</RightName>
    </Permission>
  </Permissions>
</XAccess>
```