Identity Web Services leicht gemacht

am Beispiel der TUMonline-IntegraTUM-Rückkopplung

ZKI AK Verzeichnisdienste

Dr. Ralf Ebner

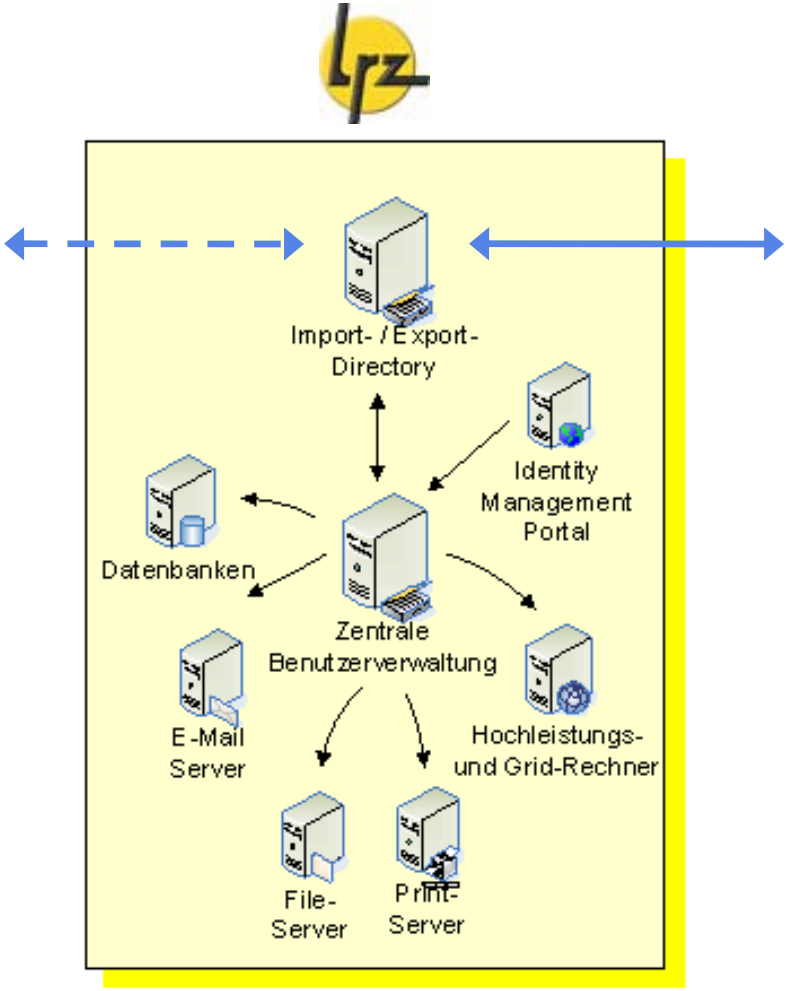
ebner@lrz.de

Agenda

- IDM im Münchener Wissenschaftsnetz
- Identity-Services für das neue TUM-Campus-Management-System
- Einfache Ad-hoc-Implementierung
- Bewertung, Vorteile, Nachteile, Probleme
- Ausblick: Architektur-Einordnung

Identity Management im Münchner Wissenschaftsnetz

x



IDM an der TUM

Identity Management bisher:

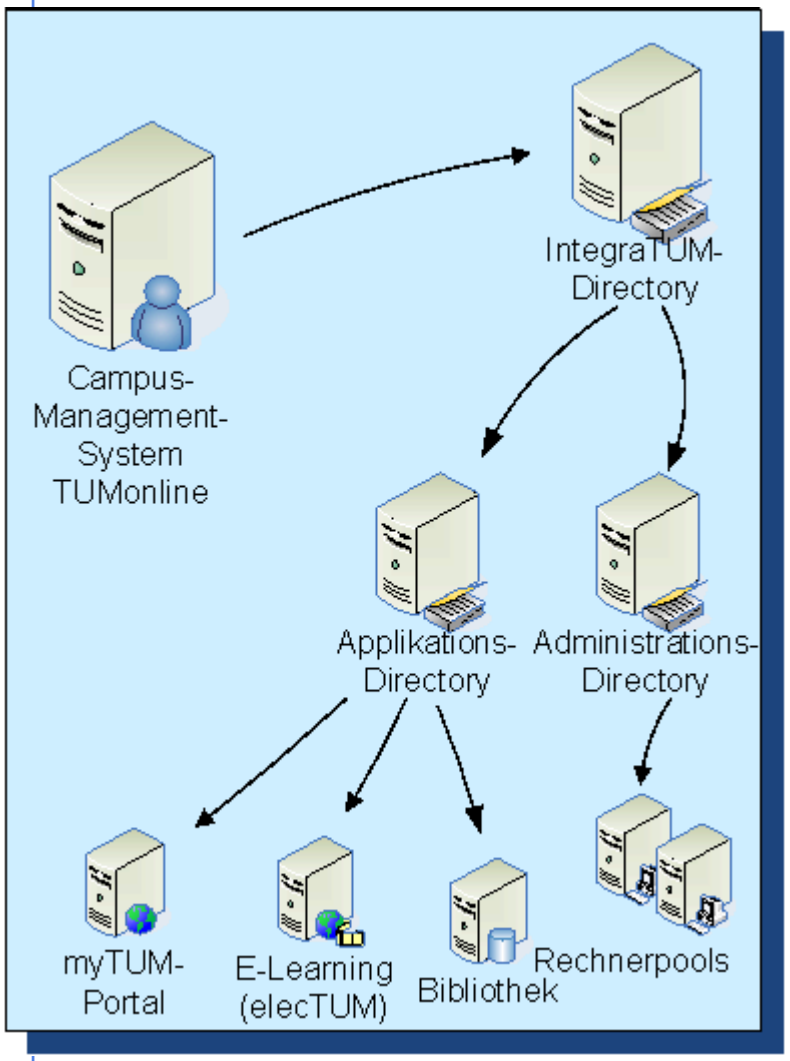
- im Verwaltungs-Directory

Ablösung der Quellsysteme:

- SAP HR via TUMonline
- HIS-SOS durch TUMonline
- Gäste durch TUMonline
- UnivIS-Rechte durch TUMonline
- Self Services in TUMonline

Identity Management zukünftig:

- in TUMonline



Identity-Services für TUMonline

Umstellung der IntegraTUM-Datenquellen auf

TUMonline als führendes System benötigt:

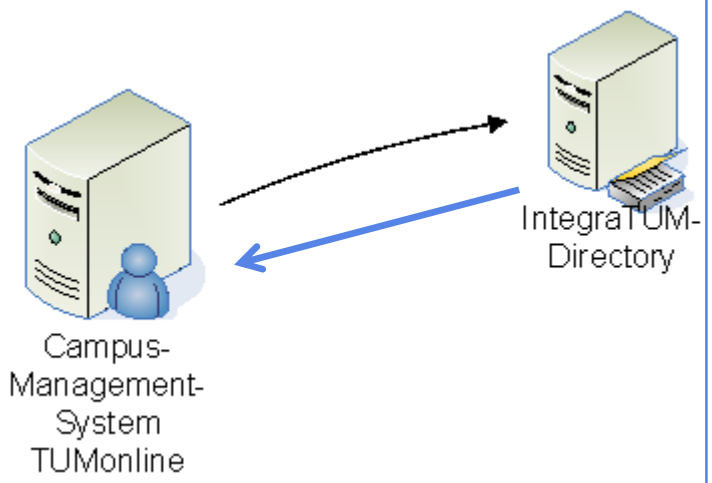
1. Bestandsdaten:
   Import von Bestandsdaten aus dem IntegraTUM-Directory
2. für Neueinträge:
   TUMonline benötigt vom LRZ:
   MWNID, LRZ-Kennung, UIDnumbers
3. zur Dubletten-Prüfung:
   Abfragemöglichkeit für TUMonline auf zukünftig größerem MWN-Datenpool (inkl. LMU-Daten)

Konzept für Identity-Services

Möglichkeit a)

Rückkanal im Novell IDM-Treiber

- Implementierungsaufwand seitens TUMonline
- Bidirektionaler Datenfluss im Novell-IDM-System
- Datenquell-Konflikte
- Steuerung nur durch Events



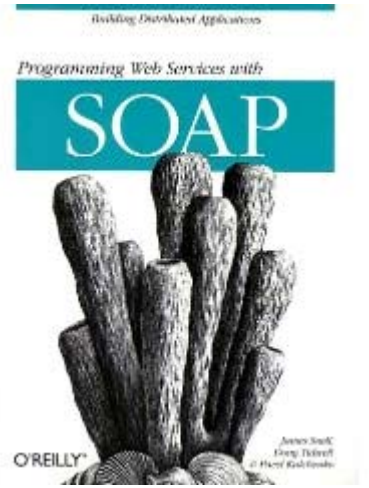
Möglichkeit b)

Funktionen als Web-Services

Ausgangslage am LRZ

- Bisher keine integrierte SOA-Infrastruktur
  - insb. kein Enterprise Service Bus (ESB)
- Perl-Module vorhanden:
  - für IDs, IDM und LDAP
- Perl-Knowhow:
  - Einsatz im Admin- und Appl.-Umfeld
  - Erfahrung und Schulungen
    => Nachhaltigkeit, Wartbarkeit
  - Projektnahe SOAP-Erfahrung
    (SOAP-Interface des LRZ-ID-Server)

→ Entscheidung für Perl auf Basis von SOAP::Lite



Vorhandene Implementierungen

Perl-Module und Schnittstellen am LRZ

- Zuteilung einer MWNID

  mwnid::getNew()

- Zuteilung einer LRZ-Kennung

  lrzkennungen::getNew('tum')

- Nachtrag von UID Numbers

  lrzkennungen::assignUID($kenn)

- Fuzzy-Suche (VNG) nach Dubletten

  VNGsearch::search($base,$vn,$nn,$geb,…)

MWN-ID-Server

LRZ-ID-Server

IntegraTUM-Verzeichnis

WS-Implementierung mit SOAP::Lite

1. SOAP::Data: Schnittstellen-Anpassung (bei Bedarf)

```
use SOAP::Transport::HTTP;
SOAP::Data->name($name=>$value)
          ->type('date')-> prefix("m");
```

2. SOAP::Transport::HTTP: Schnittstellen exportieren:

```
SOAP::Transport::HTTP::CGI
  -> dispatch_to('/usr/local/itumldap/tumonline/',
       'lrzservice::getNewMetaDirectoryID',
       'lrzservice::getNewLRZKennung',
       'lrzservice::doIdentityQuery',
       'lrzservice::doIdentityMetadataQuery
  -> handle;
```

Modul einbinden

Funktionen als Services festlegen

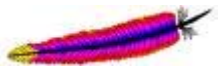
Automatische Behandlung: SOAP/HTTP-Verpackung

3. Als CGI-Skript in Webserver verfügbar machen

Zugriff und Sicherheit

Delegation an Webserver

z.B. Apache httpd



- ausschließlich HTTPS:

```
<VirtualHost *:443>
   SSLEngine on
```

- IP-Beschränkung:

```
Order deny, allow
Allow from 10.156.…
```

- evtl. einfache Authentifizierung

```
AuthType Basic
AuthUserFile …
```

Aufruf, Test und Debugging

Aktivierung zur Fehlersuche, Protokollierung, etc.

```
use SOAP::Lite +trace =>
    all => sub { print @_; } ;
```

Statt all auch z.B. method, fault, transport

Webservice-Aufruf:

```
my $soap = SOAP::Lite->
        proxy($url)->uri($uri);
@result = $soap->call($method, @wsParams)
        ->result;
```

Vorteile

1. Trennung Funktionalität – Schnittstelle
   (ähnlich einfach wie Java+Apache Axis)
2. Einfache Behandlung mehrerer Ergebnisparameter
3. Automatische oder selbst definiert Typisierung
4. Installation und Administration:
   - Standard Apache Webserver, einfach CGI-Skripte
   - keine App-Server, ESB etc. notwendig
5. Beschleunigungsmöglichkeit:
   - Berechnung: durch mod_perl
   - Transport: Load-Balancer (zustandslos)
6. Stabilität: schneller Restart des httpd, Load-Balancer

Nachteile

- Logging manuell, nicht so elegant wie Java Log4j mit AOP (aspectj)
- Keine WSDL-Generierungsmöglichkeit
  Aber: SOA überhaupt notwendig?
  → Trend zu WOA!
- Probleme:
  - HTTP-Header (bei UTF-8)
  - WSDL-Konformität (gem. TUMonline)

Problem HTTP-Header

Problem:

Content-Length bei UTF-8-Daten

Lösung:

durch Überschreiben der bytelength-Funktion:

```
sub SOAP::Utils::bytelength {
    no bytes; # Zeichencodierung ist wichtig!
    my $content = @_ ? $_[0] : $_;
    return length($content);
}
```

Problem WSDL

Problem: Namespace an gegebenes WSDL anpassen

Lösung durch zwischengeschalteten Serializer:

```
SOAP::Transport::HTTP::CGI
  -> dispatch_to('/usr/local/itumldap/tumonline/', …)
  -> serializer(MySerializer->new)
  -> handle;

package MySerializer;
@MySerializer::ISA = 'SOAP::Serializer';
sub envelope { my ($self, $header, $data) = @_;
    if ($header =~ /^(?:method|response)$/) {
        $data = SOAP::Data->name($data)->prefix('m')
                          ->uri('urn:lrzservice');          }
    return $self->SUPER::envelope($header, $data);
}
```

Architektur-Einordnung

Web Services

Web-orientiert (WOA)

= Ressourcen-orientiert

-- Meta-Directory

-- Personen-Eintrag

-- ID-Server

Service-orientiert (SOA)

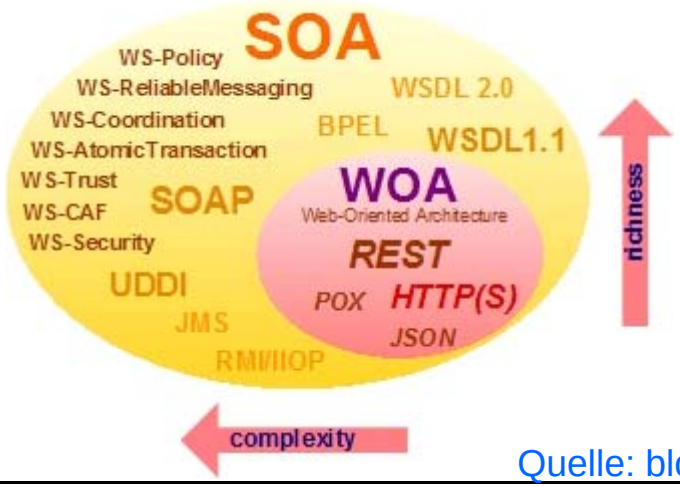
-- SOAP

-- Prozesse

-- Registrierung, Lookup

-- BPEL, Orchestrierung

-- ESB



Quelle: blogs.zdnet.com

Ausblick: WOA-Ausrichtung

- Ressourcen-orientiert (= Directory, ID-Server) [OK]
- REST (Representational State Transfer):
  zustandslos [OK]
- URI basiert: Methoden in URL und XML möglich [OK]
- XML+HTTP: SOAP eigentlich zu schwergewichtig,
  aber zur Strukturierung hilfreich [~OK]
- GET statt WSDL: Info-Seite/Service [to do]
- PUT/POST/DELETE statt ldapadd/ldapmodify
  [bei Bedarf]

Zusammenfassung

- MWN-IDM-Überblick
- Aufbau auf bestehenden Schnittstellen und Ressourcen (ID-Server, Directories)
- Web-Services für TUMonline-Daten-Rückfluss
- Einfache WS-Implementierung mit Perl SOAP::Lite + Apache Webserver
- Zielrichtung WOA