Transitioning Applications to Ontologies

Methodologies & Tools

Florence Amardeilh
Mondeca

Funded by: European Commission – 6th Framework
Project Reference: IST-2004-026460
Talk Outline

- Introduction
  - TAO project overview

- TAO Methodology
  - Detailed Revised Methodology
  - TAO cookbook

- TAO Suite
  - Ontology Learning
  - Semantic Content Augmentation
  - Semantic Web Services

- Example: Amazon web services
Addressing the problem of transitioning legacy applications to ontologies

What is a legacy software system:

“A large software system that is vital to [an] organisation, but resists modification and evolution to meet new and constantly changing business requirements”

Towards semantic-assisted software engineering
Transitioning Applications

- Legacy application:
  - database driven
  - no interoperability

- Ontologies + SOA:
  - Learn ontologies
  - Manage complex resources (multi-terminologies, multilingual) and knowledge links
  - Use Service Oriented Architecture to integrate added value services from other suppliers: cartography, translation, booking services...
Goal of TAO from the Methodology perspective

- **Support the creation of semantic web services**
  - Engineer starts with:
    - Existing legacy system
    - Domain knowledge is generated from several sources
      - Documentation generated by service provider
      - Domain information
      - User forums / blogs / community
  - Needs to generate:
    - Service / Domain Ontologies
    - Annotated Web Services Semantically
    - Annotated Documentation to support
      - User access to Services
      - Subsequent Development/Refinement

- **Challenge**
  - To develop a methodology that describes (abstractly) how to transition from existing legacy system to Semantically Rich Service Environment
Methodology Challenges

- Methodologies for supporting service developers
  - Improving Document Search
  - Annotating services

- Two individual threads are being considered:
  
1. **Generate and utilise ontology** for supporting the annotation of user documentation
   - Ontologies should support the search and retrieval (question/answer) of documentation by user/developer community

2. **Generate and utilise ontology** for supporting the annotation of services
   - Ontologies should support service-based activities (e.g. search, composition) with business/service community

- Should generate a single ontology for both
Achievements so far

- **Continual refinement** of the methodology, adapting to the other tools within the TAO suite
  - Identify usage guides as part of the methodology

- Provide a grounding for converging on a choice of ontology representation (choice of OWL-S or WSMO)

- Make the methodology more accessible
  - Cookbook-style guideline
  - Complemented with many tips and notes

- Examine **alternate recommendations** for transitioning at the service level
  - Formal approaches for transitioning legacy code
  - Evaluating intermediate representations
Overview of Methodology

Provider & Developer Communities Documents
Could be distributed amongst developers

Generating Ontologies For **Service** Annotation

1.a) Evaluate & Refine Ontology based on competency criteria
1.b) Annotate Services
1.c) Evaluate Services

SA-WSDL Description

CA/TAO Suite

HKS Distributed Knowledge Store

Ontology Learning

Generating Ontologies For **Document** Annotation

2.a) Evaluate & Refine Ontology based on competency criteria
2.b) Annotate & Query Documents

New Developer Community Documents

CA/TAO Suite
Initial Methodology Key Steps

Key Stages within Evolving Methodology
1. Source Document Identification
2. Training Data Elicitation
3. Learning the Ontology
4. Annotating...

Developer Resources
a) Supporting Annotation of user documentation

Services
a) Creating Service
b) Annotating Service
c) Evaluating resulting service

5. Ontology Evaluation
Various formats for different usages

Abstract methodology

UML diagram

Quick reference cook-book
Overview of revised Methodology

Knowledge acquisition:
- Collect textual docs for the legacy app
  - Collect reference manuals
  - Collect source code comments
  - Collect annotator’s guide
  - Collect forum discussion
  - Collect users’ guide
- Collect structure docs for the legacy app
  - Collect source code
  - Collect WSDL definitions
  - Collect APIs
  - Collect XML data
  - Collect database schema

Document Corpora:
- Save corpora into TAO Repository

Ontology Learning:
- Identify structure and content software artifacts
- Transform content and structure into feature vectors
- Create domain ontology from feature vectors
  - Create Concept (Unsupervised)
  - Create Concept (Supervised)
  - Manage concepts
  - Manage relations
  - Manage instances
- Ensure the ontology is well-formed
- Ensure the ontology has right formalism
- Ensure the ontology is consistent
- Evaluate and refine Ontology

Domain Ontology:
- Identify other contents for annotating
  - Yes
  - Identify services
  - WSDL Definitions
  - Documents for annotating
- Domain Ontology

Service and content augmentation:
- Annotate services and other contents using WDP3
  - Load/Import WSDL or other resources
  - Load domain ontology
  - Start annotating
  - Manually annotating
  - Automatically annotating
  - View and revise annotations
  - Ontology population
- SA-WSDL
- SA-Doc
- Save SA-WSDL and SA-Doc into TAO repository
- Evaluation and refine service descriptions

Activity supported by HKS (WP4)
Activity supported by LATINO (WP2)
Activity supported by GA MANAGER (WP3)
Overview of revised Methodology

Knowledge Acquisition

2. Documents and service definitions are identified, including:
   - Reference Manuals
   - Forum Discussions
   - Source Code
   - WSDL & API defns
   - DB & XML schemas

3. Produce a corpora in the TAO repository
Overview of revised Methodology

Ontology Learning

2. Identify textual & service artifacts:
   - Construct Feature Vectors
   - Map resources to training instances

3. Identify concept clusters and weightings

4. Generate & Refine ontology, ensuring:
   - Consistency
   - Good formalism
   - Context relevance
Overview of revised Methodology

Service and Document Annotation

• Construct WSDL documents (if necessary)
• Annotate using CA / Gate tools
  • Load ontology & WSDL / document resources
  • Automated initial annotation
  • Refine / revise annotations
  • Use other CA tools to refine ontologies
• Generate SA-WSDL from annotated WSDL description
Converging on SA-WSDL

- Converged on using SA-WSDL for semantically annotating Web Services
  1. Neutral with respect to current ontology representation languages (RDF, OWL, WSML,...)
  2. SA-WSDL adequately fits with TAO requirements better than WSMO and OWL-S frameworks
  3. Choreography and Orchestration issues are being put aside (focus is on ontological issues)
  4. Convergence on a single ontology for both service and documentation annotation
Motivating the CookBook

- Resulting Methodology needs to be communicated to practitioners in a usable way

- Using the cookbook metaphor
  - Several examples given with step-by-step instructions
  - Each example illustrates a different challenge problem
The **Cookbook** metaphor has been taken from:

- **PostScript Language Tutorial and Cookbook**

Include a set of walk-through examples using the different tools of the TAO suite

- *Illustrates parameter settings, steps in using tools, and expected results.*
- *User should be able to reproduce the generated ontologies themselves*

To cover:

- **Amazon Services**
- **Gate Services**

Also as a Web-based guide (Wizard)
Overview of Methodology

TASK 1. Create a new project in TAO Suite
TASK 2. Knowledge acquisition
TASK 3. Save the document corpuses to TAO Repository
TASK 4. Ontology learning
  4.1 Identify content and structure of software artifacts
  4.2 Transform contents and structures into feature vectors
  4.3 Create domain ontology from feature vectors
  4.4 Design Ontology
  4.5 Save domain ontology into TAO Repository
TASK 5  Service and content augmentation
  5.1 Identify the contents to be annotated
  5.2 Load the legacy contents and ontology
  5.3 Start Annotating
  5.4 Revise annotations and ontology (Section 4.5.4)
    5.5 Ontology population
  5.6 Save Annotations
    5.7 Deploy, evaluate and refine services
Many tips and notes to help users make choice

Tips:
- To help users develop from APIs, it is important to understand what textual document should be assigned to each text element.
- Java/C++ support should be used to analyze potential candidates.
- Class, objects, and methods should be used to create an ontology.

Tips:
- For Java/C++ classes, the potential candidates can be derived from an existing database.
- Inheritance and interfaces should be considered.

Tips:
- The following gives some general rules for users to decide if "unsupervised" or "supervised" should be adopted.
- "Supervised" approach is intended for the cases where the user has a clear idea of the sub-concept he wants to add to the ontology but the unsupervised methods do not discover it.
- If users already had a primary version of ontology (either gotten from a database or directly)

Tips:
- Most ontology editors, including TAO Suite, can be used to check that the ontology is well-formed. Users can also choose some ontology development environments which they are familiar with, such as Protégé and TopBraid Composer.
Quick references to the major tasks

| TASK 1. Create a new project in TAO Suite (Section 4.1) |
| TASK 2. Knowledge acquisition (Section 4.2) |
| TASK 3. Save the document corpuses to TAO Repository (Section 4.3) |
| TASK 4. Ontology learning (Section 4.4) |
|   Task 4.1 Identify content and structure of software artifacts (Section 4.4.1) |
|   Task 4.2 Transform contents and structures into feature vectors (Section 4.4.2) |
|   Task 4.3 Create domain ontology from feature vectors (Section 4.4.3) |
|   Task 4.4 Design Ontology (Section 4.4.4) |
|   Task 4.5 Save domain ontology into TAO Repository (Section 4.4.5) |
| TASK 5. Service and content augmentation (Section 4.5) |
|   Task 5.1 Identify the contents to be annotated (Section 4.5.1) |
|   Task 5.2 Load the legacy contents and ontology (Section 4.5.2) |
|   Task 5.3 Start Annotating (Section 4.5.3) |
|   Task 5.4 Revise annotations and ontology (Section 4.5.4) |
|   Task 5.5 Ontology population (Section 4.5.5) |
|   Task 5.6 Save Annotations (Section 4.5.5) |
|   Task 5.7 Deploy, evaluate and refine services (Section 4.5.6) |
The TAO web-based wizard

♦ Demo of TAO Suite wizard
TAO Suite & Methodology

- A comprehensive support of the TAO methodology

Demo available on TAO web site
TAO Suite – putting it all together

- A complete transitioning environment
- Manage all transitioning processes and tools within a uniform interface
TAO Suite Components

- Architecture as integration mechanism
  - Ontology generation
    - TAO Suite
    - LATINO
    - HKS
    - Exporting/Importing with external GUI tools

- Content Augmentation
  - TAO Suite
  - CA MANAGER
  - HKS
  - Exporting/Importing with external GUI tools

- Creation of SWS descriptions
  - TAO Suite
  - CA MANAGER
  - HKS
  - Exporting/Importing with external GUI tools
TAO Suite - architecture

Components integration and communication in TAO Suite

- LATINO/OntoGen
- RDBToOnto
- HKS
- CA Manager
- Annotation UI
- GATE CA Services

TAO Suite
Achievements so far

- Architecture and integration requirements and specifications

- Functional aspects of TAO Suite:
  - Integrating into an IDE the TAO components:
    - LATINO
    - CA MANAGER
    - HKS
  - Interacting with TAO components for performing a first automatic generation of semantic-based system
  - Exporting/Importing the generated semantic descriptions (user can manually refine them)
  - Whole transitioning process supported by TAO Suite is semi-automatic
TAO Suite objectives
TAO Suite Architecture

General architecture
TAO Suite Architecture

General architecture
Demo of TAO Suite pre-prototype
TAO Suite - features

Authentication user/project
TAO Suite - features

New project creation
TAO Suite - features

General persistency layer. Support HKS, MySQL, Sesame…
TAO Suite - features

Quick access to external tools

Runtime hot deployment
TAO Suite - features

Adapters for CA Manager
TAO Suite - features

Different adapters for creation of SWS
TAO Suite - features
Annotation User Interface

Demo of Annot UI Prototype
TAO Suite – Downloads

Demos and Downloads

Ontology Learning
Software: Latino/OntoGen; OntoSight
Videos: TAO Jan08; ESWC08
Demos: Latino/OntoGen; OntoSight; OL for Aeronautics; OL for VideoLectures.net

Content Augmentation
Software: Content Augmentation
Videos: ESWC08
Demos: OntoAnnotator; CA Manager; ITM; Annotation UI; Core Search Demo

Heterogeneous Knowledge Store
Software: Heterogeneous Knowledge Store; Distributed Knowledge Store
Videos: ESWC08
Demos: Heterogeneous Knowledge Store

Intelligent Knowledge Access
Software: Software Coming Soon
Videos: TAO Jan08; ESWC08
Demos: Querying ontologies using Natural Language; GATE Case Study Prototype - Introduction; GATE Case Study Prototype - Tutorial; Core Search Demo

Databases to Ontologies
Software: RDBToOnto
Videos: ESWC08
Demos: RDBToOnto Demo1; RDBToOnto Demo2
TAO Suite – Downloads

Aircraft Maintenance Case Study
Software: Confidential Prototype
Demos: Ac Maintenance Case Study

TAO Suite
Software: TAO Suite
Demos: TAO Suite

GATE Data
GATE Code and Documentation
GATE Web services

Amazon Case study
Demos: Amazon Case Study

Other Related Materials

Tutorials
WSMO tutorial
ESWC’08 Tutorial
Video Tutorials
SWES’08 Workshop

"Call 3 in Motion" workshop at EC

TAO demos
TAO Publicity Materials
TAO Public Deliverables
TAO Project presentation
TAO Project Demo
Questions? Discussion welcome!
Amazon Service Introduction

◊ Provides developers with direct access to Amazon's robust technology platform.
  ◊ Built from Amazon's suite of web services.

◊ 11 solutions catalog.
  ◊ Associates Web Service (formerly Amazon E-Commerce Service (ECS))
  ◊ Simple Queue Service
  ◊ Simple Storage Service
  ◊ Flexible Payments Service
  ◊ … …

◊ WP1 has generated:
  ◊ Amazon ECS ontology
    ◊ Nearly 300 named classes
    ◊ 700 Properties
Amazon Associates Web Service

- Exposes Amazon's product data
- For developers to applications that merchandise Amazon products.
- Development environment
  - Complete API and Java library
  - Developer documents,
    - e.g. tutorial, Code samples, Developer Forums
- Based on WSDL
Demonstration of the Methodology using the Amazon Services

Illustrates:

- **The ontology learning process** using OntoSight and OntoGen over Amazon data (*screenshots*)
- **The annotation process** using Gate (*video*)
Assemble Learning Resources

The relevant reference text is located and used to build learning resources. These can be visualised by the Ontosight tool to explore and identify salient conceptual clusters.
Visualise the Network

The conceptual clusters can then be visualised to identify the links between concepts prior to ontology creation. Links between clusters and concepts can be weighted.
Clusters corresponding to different parts of the service description can then be identified:

- Service Names
- Service Messages
- Data/Concept Types
- Domain Concepts
The visualization of a semantic space and the use of the context tool for exploring the space

Clusters corresponding to different parts of the service description can then be identified:

- Service Names
- Service Messages
- Data/Concept Types
- Domain Concepts
The visualization of a semantic space and the use of the context tool for exploring the space

Clusters corresponding to different parts of the service description can then be identified:

Service Names
Service Messages
Data/Concept Types
Domain Concepts
The visualization of a semantic space and the use of the context tool for exploring the space

Clusters corresponding to different parts of the service description can then be identified:

- Service Names
- Service Messages
- Data/Concept Types
- Domain Concepts
Load Feature Vector to OntoGen and create ontology

Based on the identified clusters, a taxonomic ontology can be generated as an OWL file.
Amazon ECS Ontology

- Amazon ECS ontology
  - Nearly 300 named classes
  - 700 Properties
- Annotate Amazon Documents

![Image of GATE 4.0 build 2823 interface with Amazon JavaDoc and Amazon Code tabs selected.](image-url)
Questions? Discussion welcome!
Achieved in second year: CA-Manager specifications

- CA-Manager: middleware used by TAO suite that connects text-mining, knowledge repository, + other components if needed

- Must be able to address:
  - **Automatic metadata generation**
    - Generate a set of metadata about the processed document
    - Ex: create a searchable database of document metadata (life sciences industry)
  
  - **Automatic knowledge extraction**
    - Populate an ontology with extracted entities
    - Ex: populate a competitive intelligence knowledge base (publishing)
  
  - **Automatic text annotation**
    - Insert specific metadata at particular offsets in the original document
    - Ex: generating SA-WSDL from original WSDL files + text-mining

- Must be able to integrate with ontology aware text-mining AND legacy text-mining tools
Text Mining

Audio Mining

Input documents

Powered by

Structured information

Ontology and vocabulary control

Reasoning engines

Metadata generation

Knowledge population (knowledge store integration)

User validation

Format transformation

XML
Achieved in second year: CA-Manager architecture

Choice of UIMA as CA-Manager backbone.
- IBM open-source framework
- « UIMA supports the development, discovery, composition and deployment of multi-modal analytics for the analysis of unstructured information and its integration with search technologies »

Benefits:
- Ability to define flexible processing flows
- Distributed & pluggable components
- XML configuration of components
- A common data structure between components
- Sharing & reusing open-source components

Other solution: 100% webservices
- No need to have webservices inside internal components of TAO
- Potentially slow & hard to maintain
- Lots of things to reimplement
Achieved in second year: CA-Manager architecture

Transistioning text mining to semantic and ontology based tools

Content Augmentation Service Type System

UIMA

CA-Manager Service

UIMA (WebService)

TS traductor

AE

Type System XYZ

UIMA

Client API

CLIENT APP

EJB

Web service

AE

CAS

... CAS

AE

CAS

AE

CAS

Regexp

Gate

OWLIM

HKS

RDF

Split → Extract → Merge → Match → Control → Infer → Store → Serialize → ...

UIMA

HKS

RDF

Regexp

Gate

Various tools and systems in the pipeline:
Achieved in second year: CA-Manager architecture

◊ The architecture
  ◊ Is a specialized UIMA pipeline
  ◊ Focuses on transitioning text-mining results to semantic tools
  ◊ Is meant to hide UIMA complexity if not required
  ◊ Provides a _service_ (simple, remote)
  ◊ Benefits from UIMA framework (process configuration, distributed architecture, connection to other UIMA based tools, etc.)
  ◊ Is pluggable (possibility to write a connector to a new external tool)

◊ Added value (how is it better than just UIMA?)
  ◊ Definition of a generic Common Analysis Structure
  ◊ Mapping language to map IE results to CAS
  ◊ Specialization of UIMA pipeline
  ◊ Connectors to various external tools
  ◊ Hide UIMA complexity
Achieved in second year: CA-Manager

✦ Demo of CA-Manager
CA-Manager : Coming next

◊ OK so far :
  ◊ Specifications and architecture
  ◊ KCIT integration
  ◊ Working prototype

◊ Coming next :
  ◊ HKS integration
  ◊ Ontology-based controls
  ◊ SA-WSDL generation
  ◊ Integration with annotation GUI
  ◊ Open-sourcing