



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

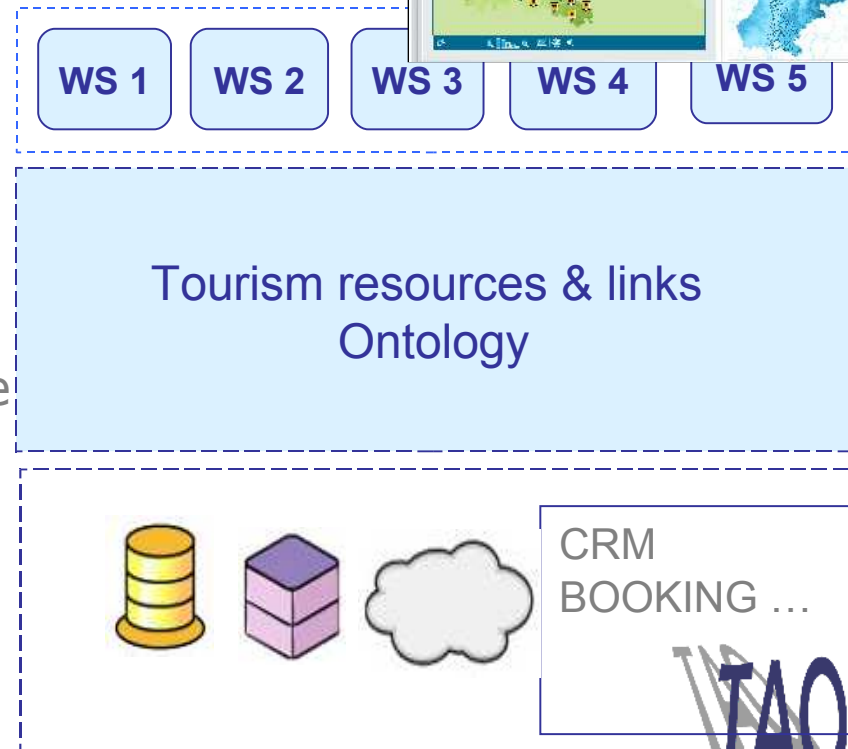


TAO project overview

- ◆ 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*

Ontology
Learning



1.c) *Evaluate
Services*



CA/TAO
Suite

1.b) *Annotate
Services*

SA-WSDL
Description



Generating Ontologies For **Document** Annotation

2.a) *Evaluate & Refine
Ontology based on
competency criteria*

Ontology
Learning



CA/TAO
Suite

2.b) *Annotate & Query
Documents*

New Developer
Community
Documents

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

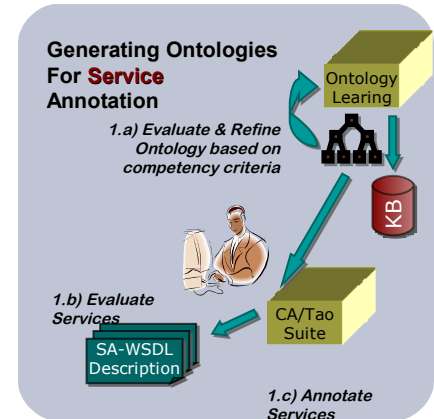
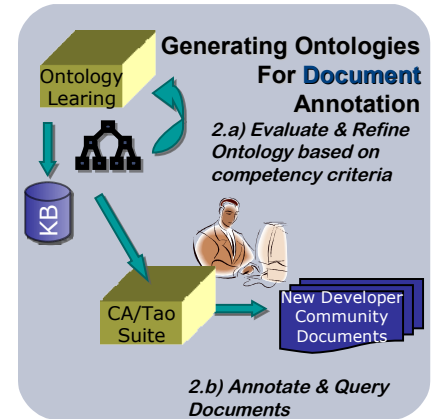
◆ Services

a) Creating Service

b) Annotating Service

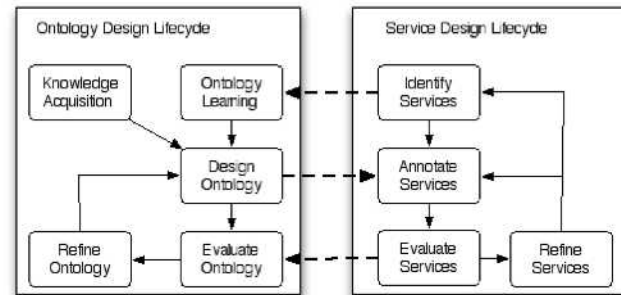
c) Evaluating resulting service

5. Ontology Evaluation

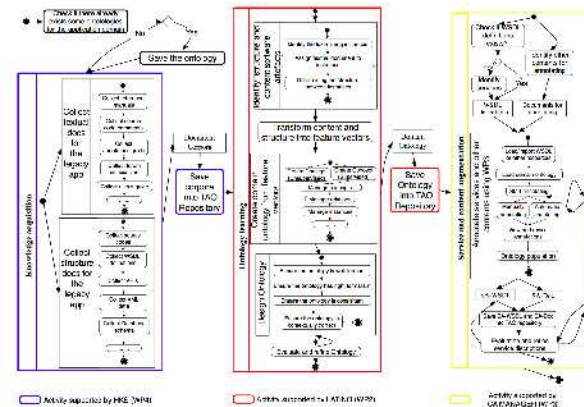


Various formats for different usages

Abstract methodology



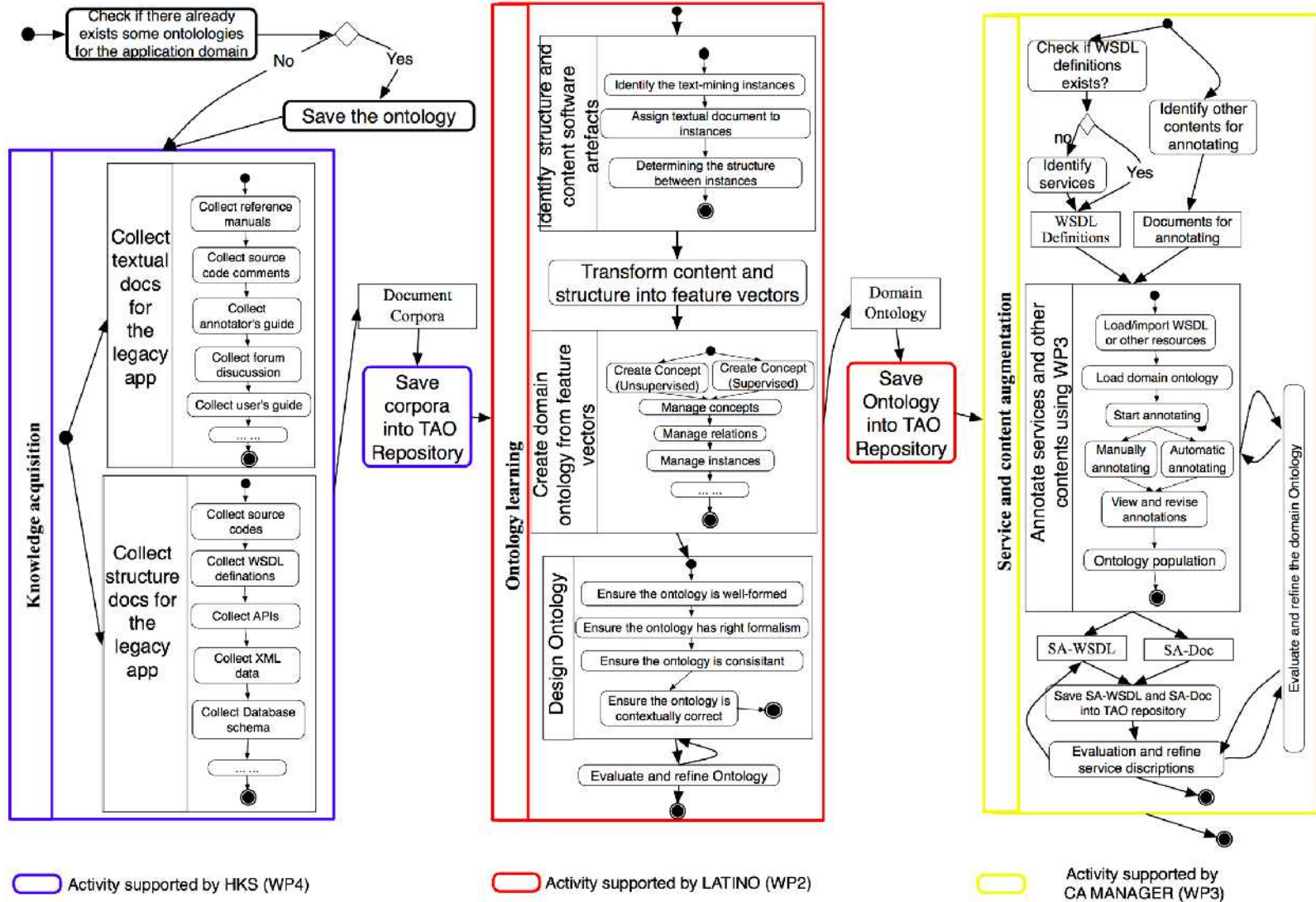
UML diagram



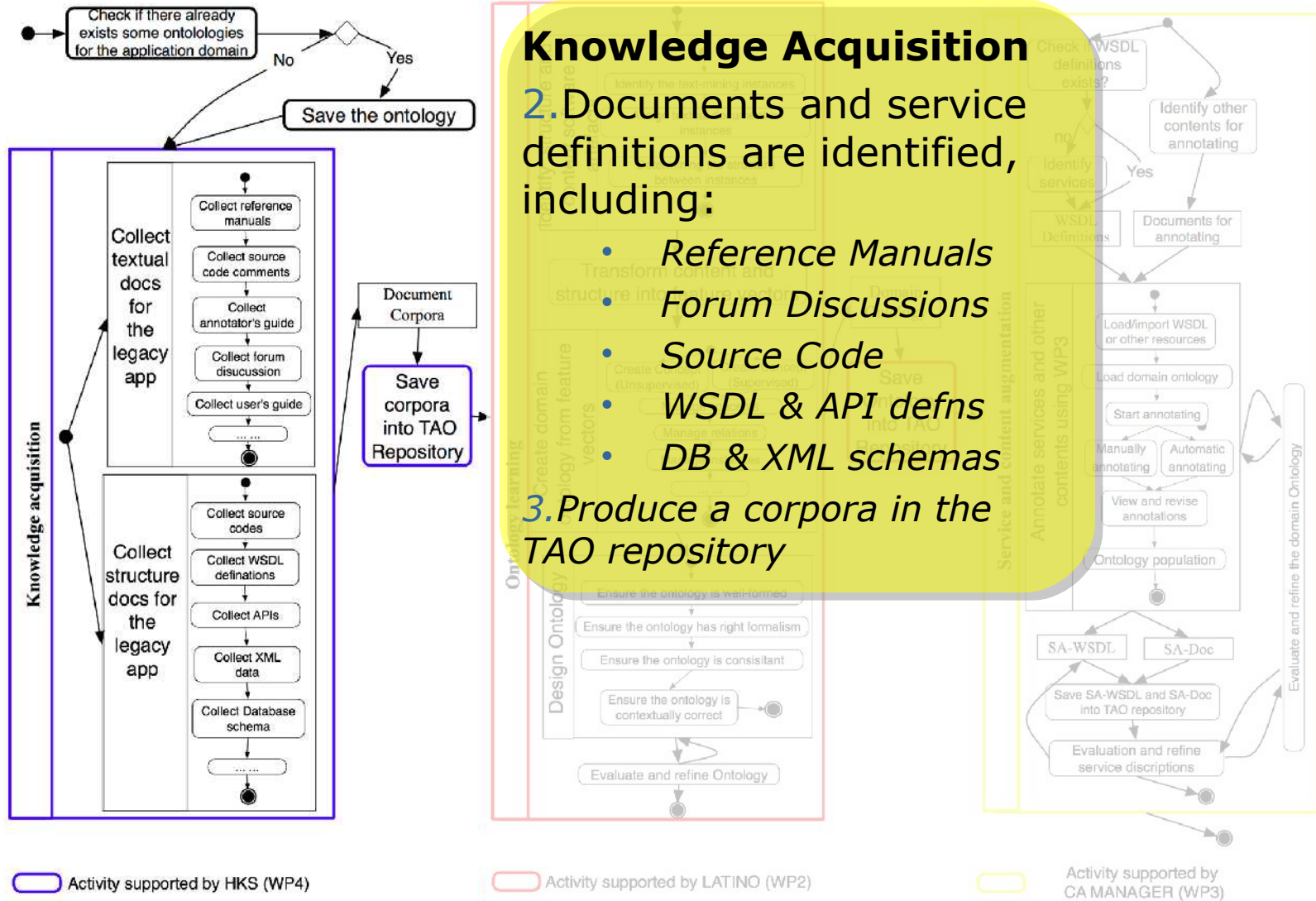
Quick reference cook-book

- 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)
 - 4.1 Identify content and structure of software artifacts (Section 4.4.1)
 - 4.2 Transform contents and structures into feature vectors (Section 4.4.2)
 - 4.3 Create domain ontology from feature vectors (Section 4.4.3)
 - 4.4 Design Ontology (Section 4.4.4)
 - 4.5 Save domain ontology into TAO Repository (Section 4.4.5)
- TASK 5. Service and content augmentation (Section 4.5)
 - 5.1 Identify the contents to be annotated (Section 4.5.1)
 - 5.2 Load the legacy contents and ontology (Section 4.5.2)
 - 5.3 Start Annotating (Section 4.5.3)
 - 5.4 Revise annotations and ontology (Section 4.5.4)
 - 5.5 Ontology population (Section 4.5.5)
 - 5.6 Save Annotations (Section 4.5.6)
 - 5.7 Deploy, evaluate and refine services (Section 4.5.7)

Overview of revised Methodology



Overview of revised Methodology



Overview of revised Methodology

Ontology Learning

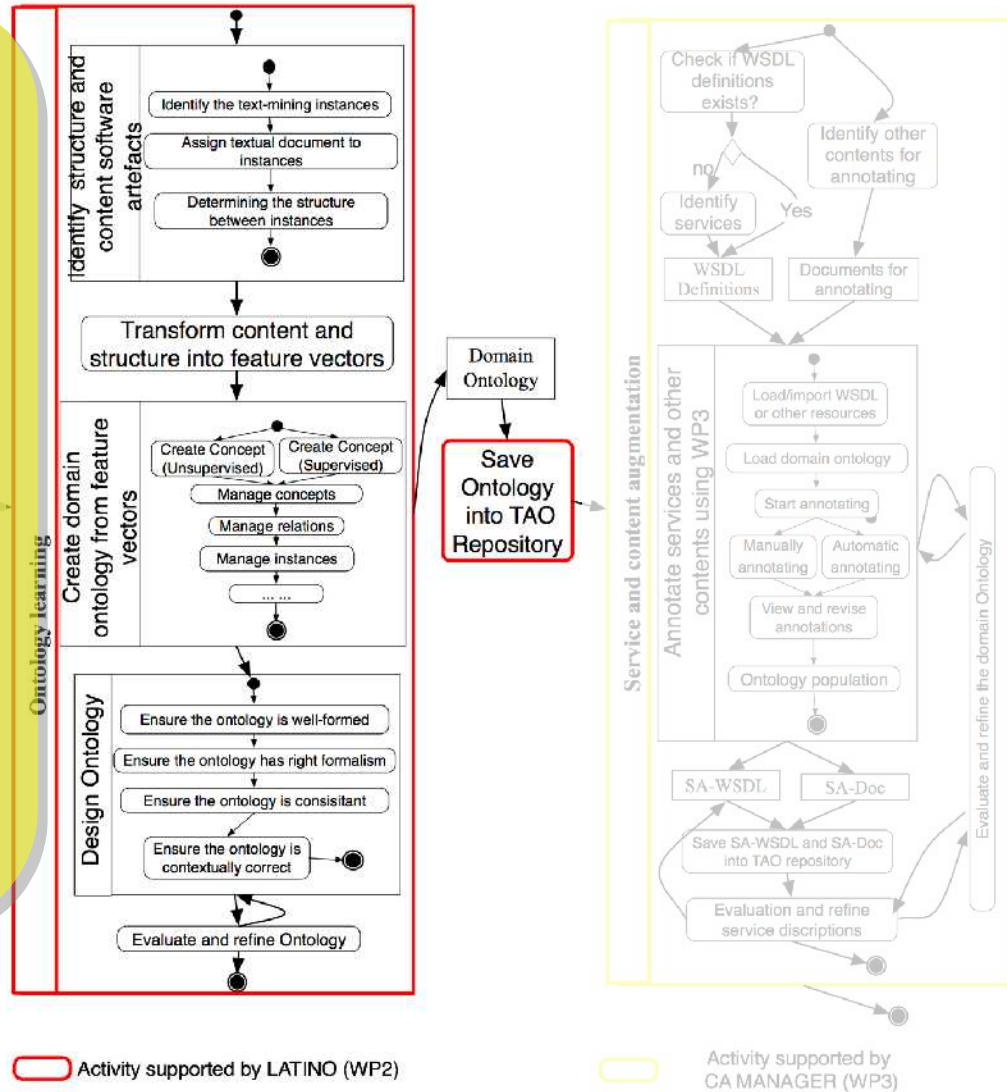
2. Identify textual & service artifacts:

- Construct Feature Vectors
- Map resources to training instances

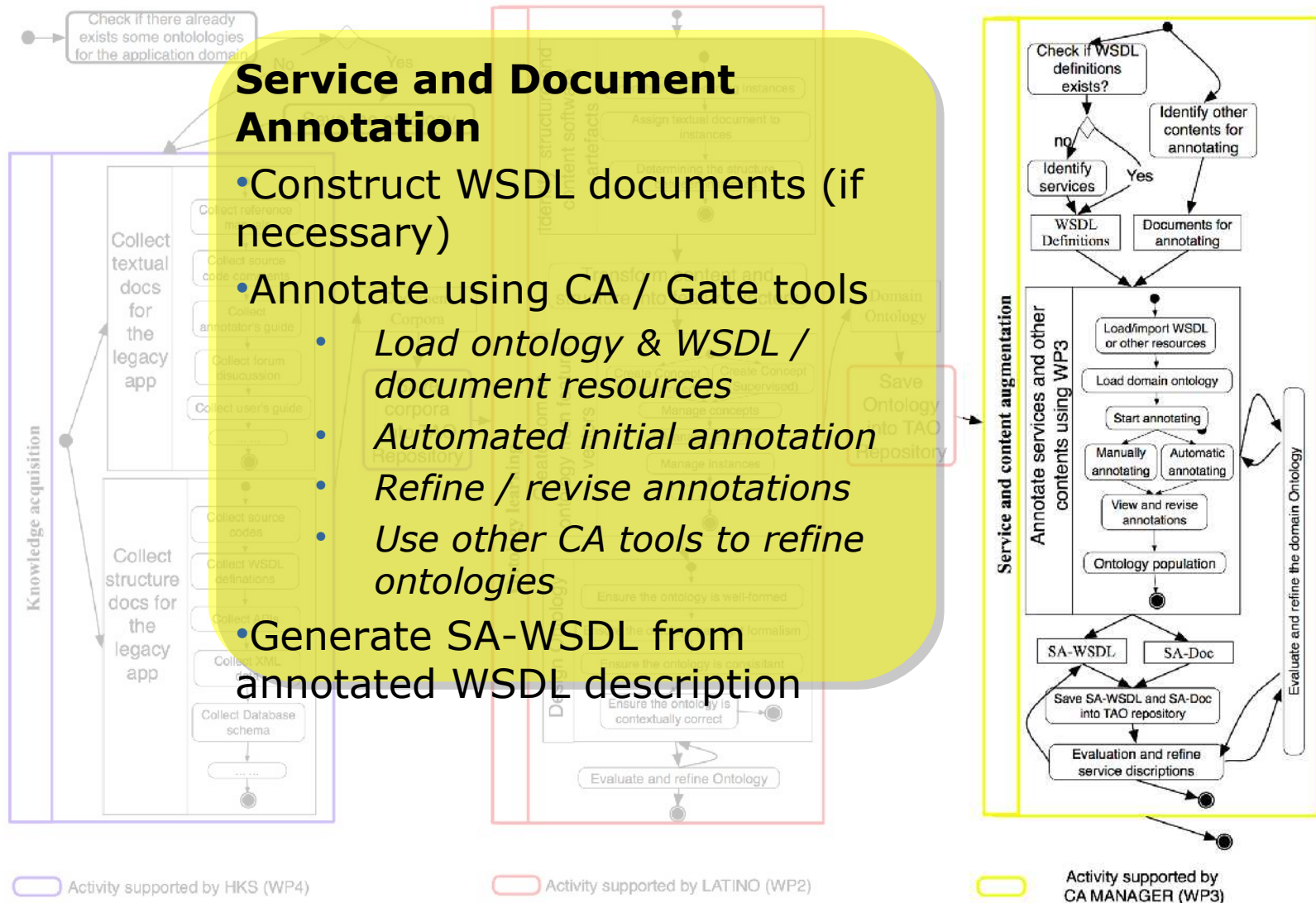
3. Identify concept clusters and weightings

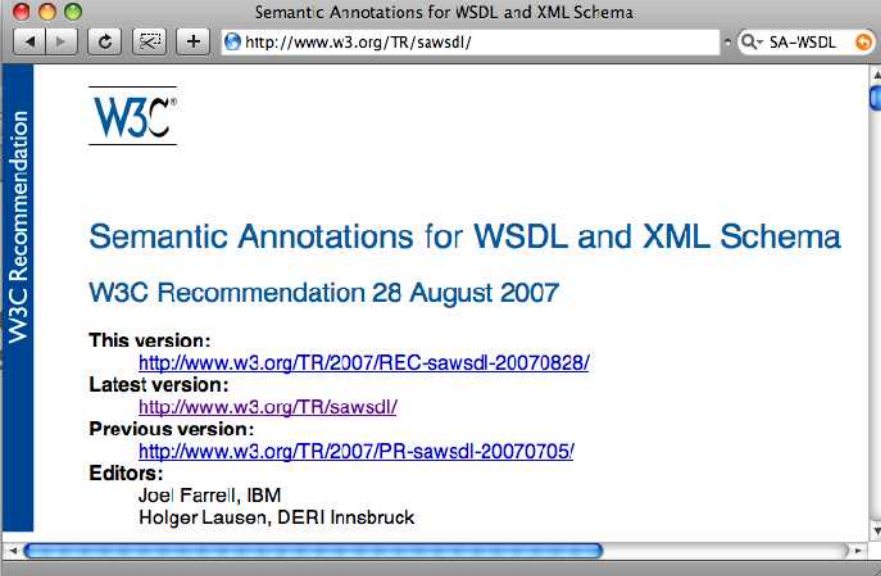
4. Generate & Refine ontology, ensuring:

- Consistency
- Good formalism
- Context relevance



Overview of revised Methodology





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

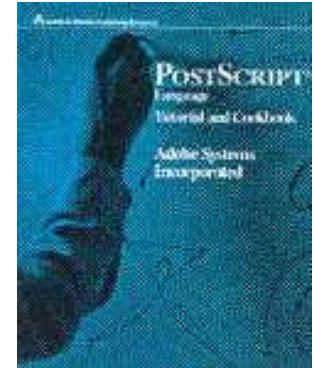
Motivating the CookBook

- ◆ 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

Cookbook Design



- ◆ 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)*

Includes:

- ◆ *Steps in gathering text sources (assets)*
- ◆ *Steps in preparing assets*
- ◆ *Determining parameters and objectives using tools*
- ◆ *Using tools for annotating Documents and SA-WSDL service*

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 annoated

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

More accessible

- ◆ Many tips and notes to help users make choice

Note:

Normally the first step in creating a Web service is to design and implement it.

including testing, which service from derivation (API), design, Web, Web, and us de to

Tips:

To help users

wh

div

str

sof

we

Tips:

It is important to consider general potential

Tips:

For Java/C++ classes, the potential

- Inheritance and interfaces
- Type references
- Class, enumeration, and interface
- Comments

Tips:

There is no common rules as what textual document should be assigned

to each text

be used to a

Java/C++ s

• class

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 file or database)

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

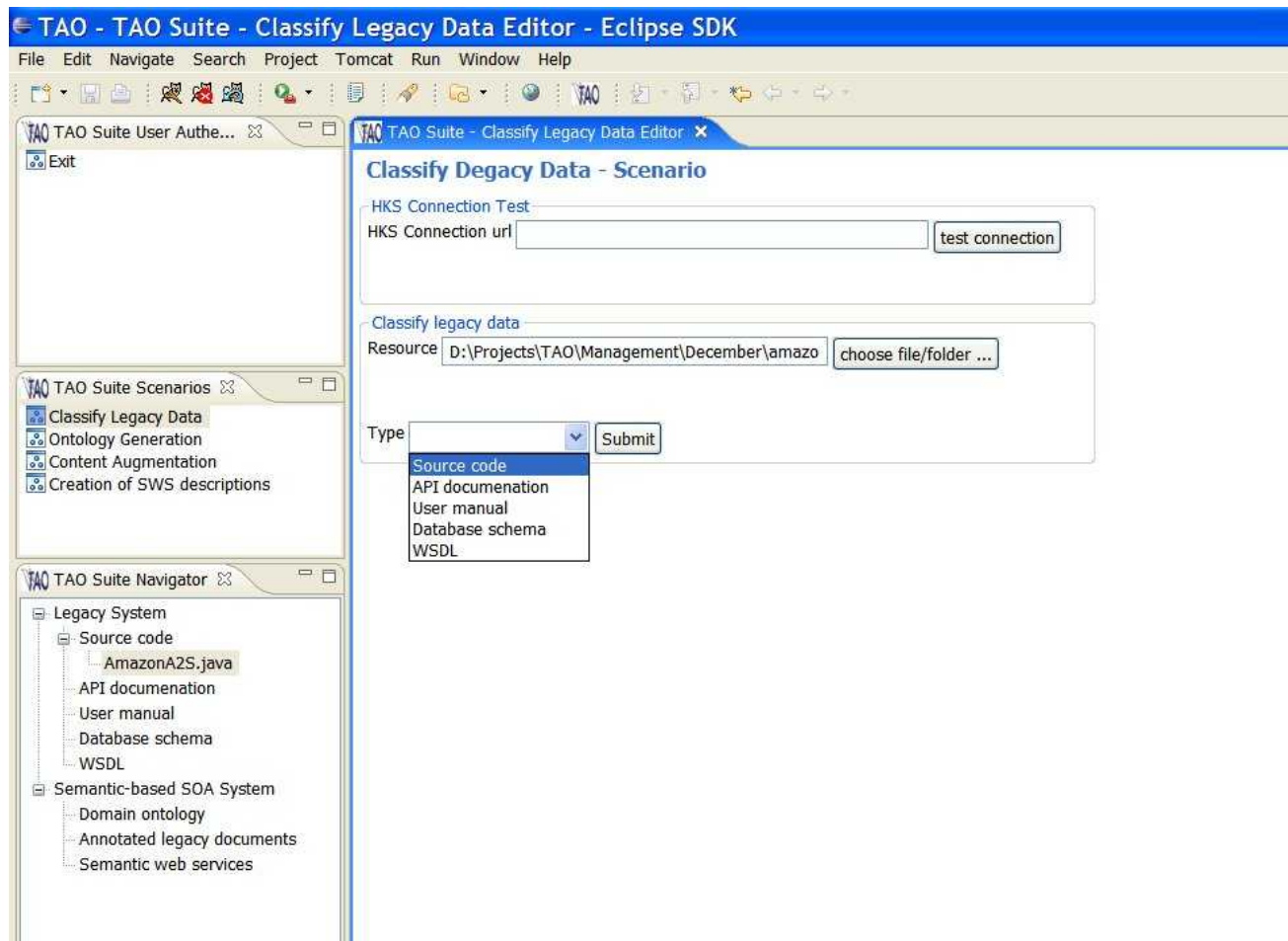
- A comprehensive support of the TAO methodology

The screenshot shows the TAO Suite - Ontology Generation Editor interface. The main window displays the 'Ontology Generation - Scenario' configuration page. A large red circle highlights the central configuration area, which includes fields for 'Latino WS Connection test', 'Latino WS connection UR.' (with a 'Test WS' button), 'Latino adapters' (with 'Existing adapters' and 'New adapter' sections), and 'Create cocurrent network' options. A light blue box on the left, labeled 'Methodology steps', points to a list of tasks: 'Classify Legacy Data', 'Ontology Generation', 'Consent Augmentation', and 'Creation of SWS descriptions'. A light yellow box, labeled 'Legacy Data', points to a tree view in the 'TAO Suite Navigator' showing a 'Legacy System' with sub-items like 'Source code', 'API documentation', and 'Database schema'. A light green box at the bottom, labeled 'Semantic SOA System', points to a tree view showing 'Semantic based SCA System' with sub-items like 'Domain ontology' and 'Annotated legacy documents'. A light pink box on the right, labeled 'TAO Tools', points to the 'TAO Suite - Ontology Generation Editor' window.



TAO Suite – putting it all together

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

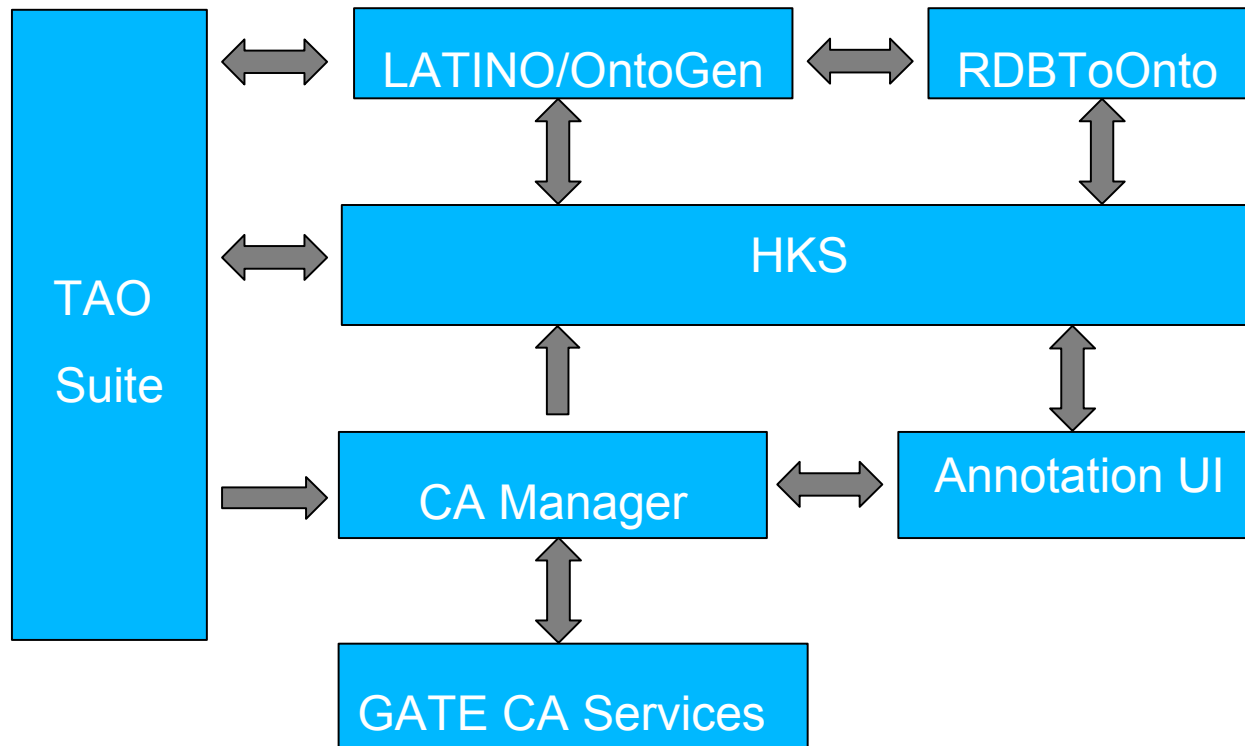


TAO Suite Components

- ◇ 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

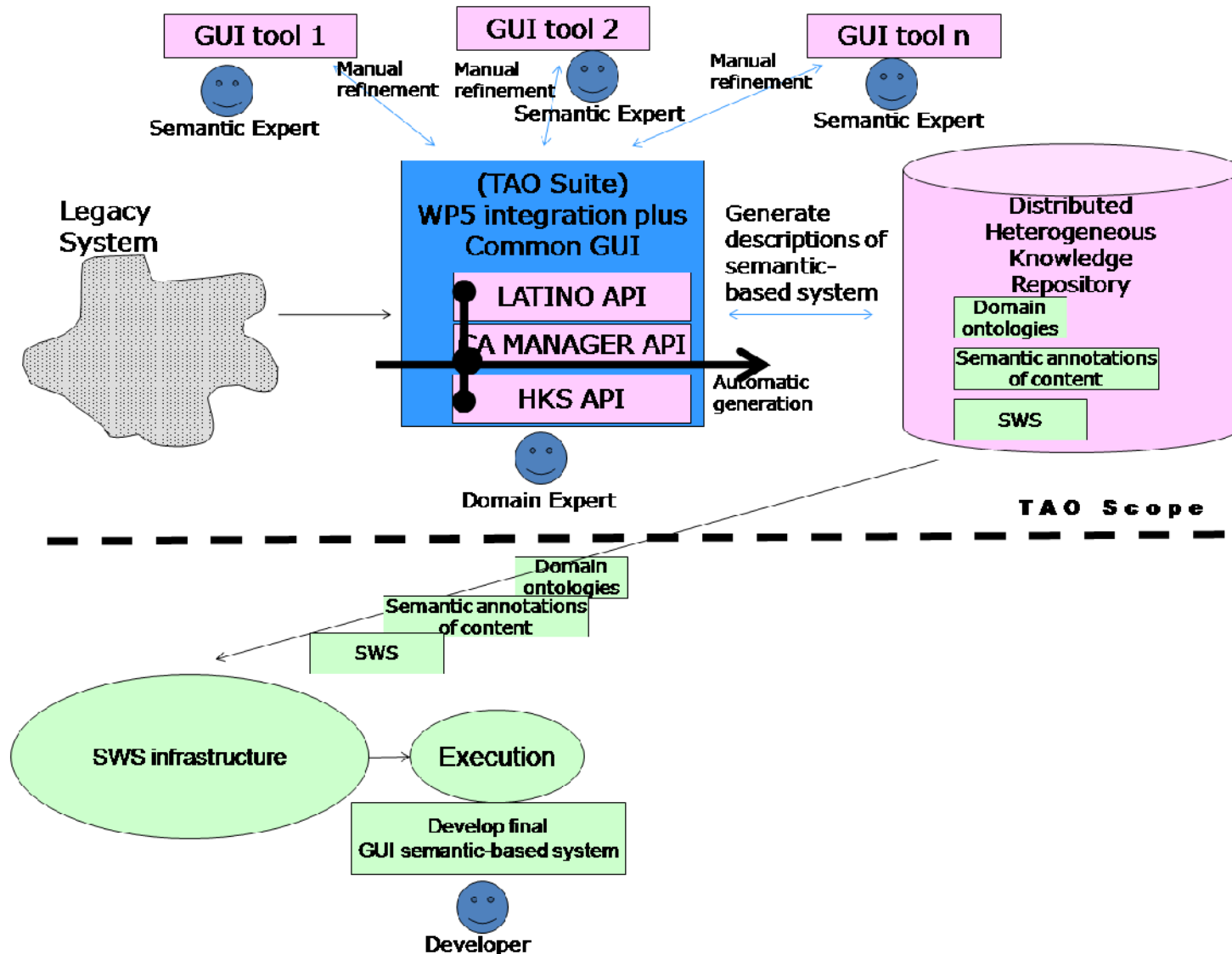
- ◆ Components integration and communication in TAO Suite



Achievements so far

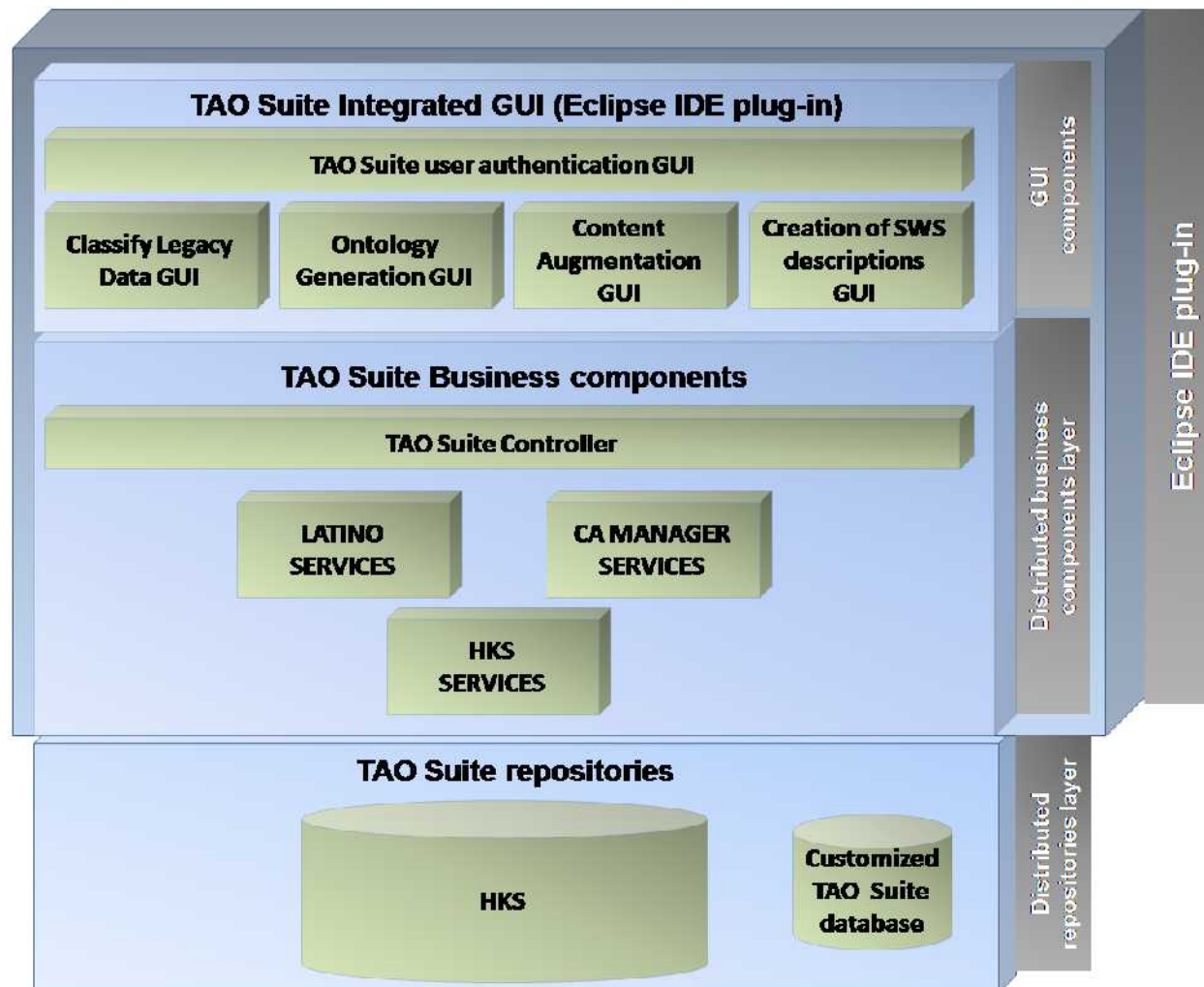
- ◆ Architecture and integration requirements and specifications
- ◆ Functional aspects of TAO Suite:
 - ◆ Integrating into an IDE the TAO components:
 - ◆ LATINO
 - ◆ CA MANAGER
 - ◆ HKS
 - ◆ Interacting with TAO components for performing a first automatic generation of semantic-based system
 - ◆ Exporting/Importing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

TAO Suite objectives



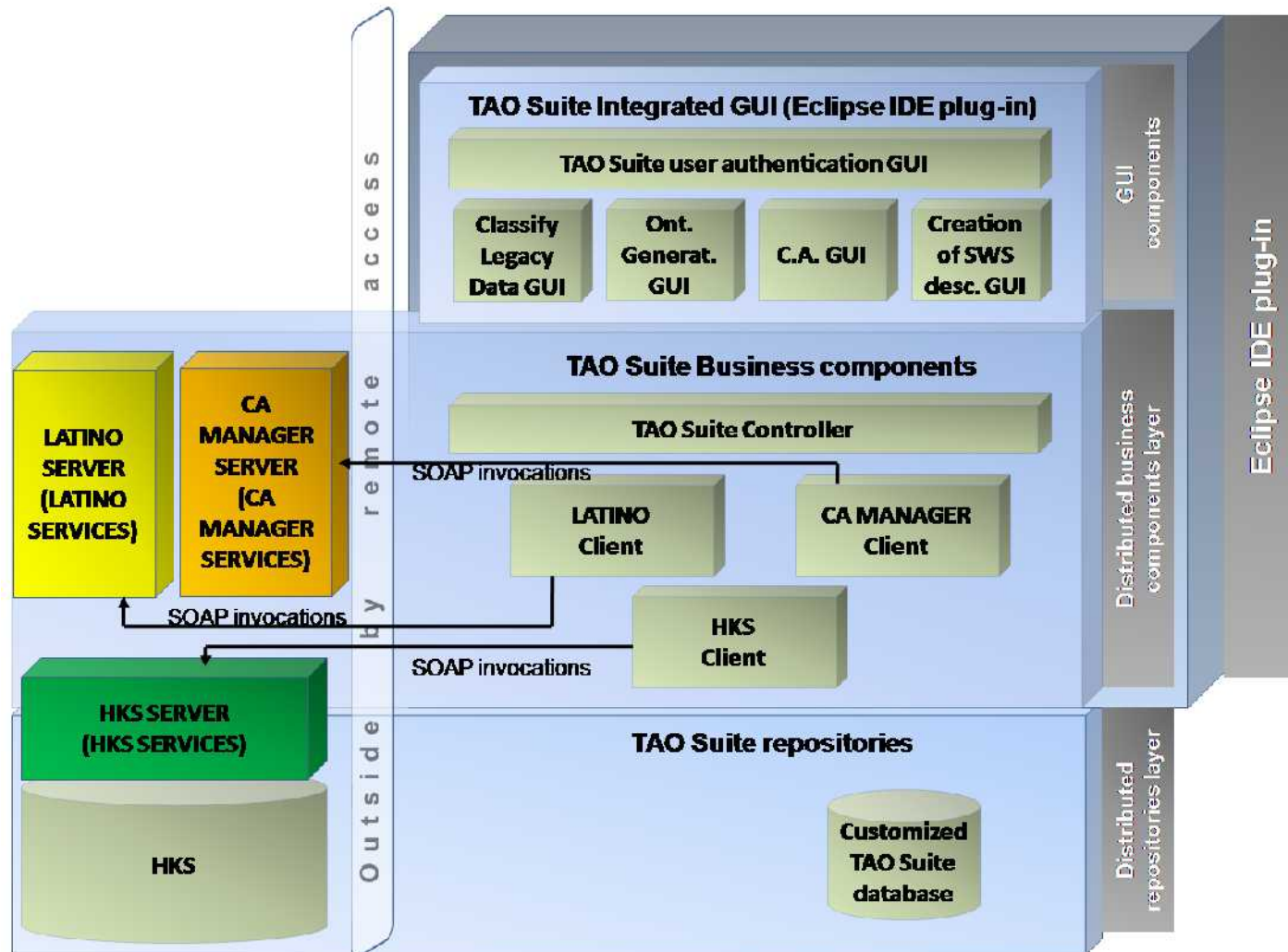
TAO Suite Architecture

◇ General architecture



TAO Suite Architecture

◇ General architecture

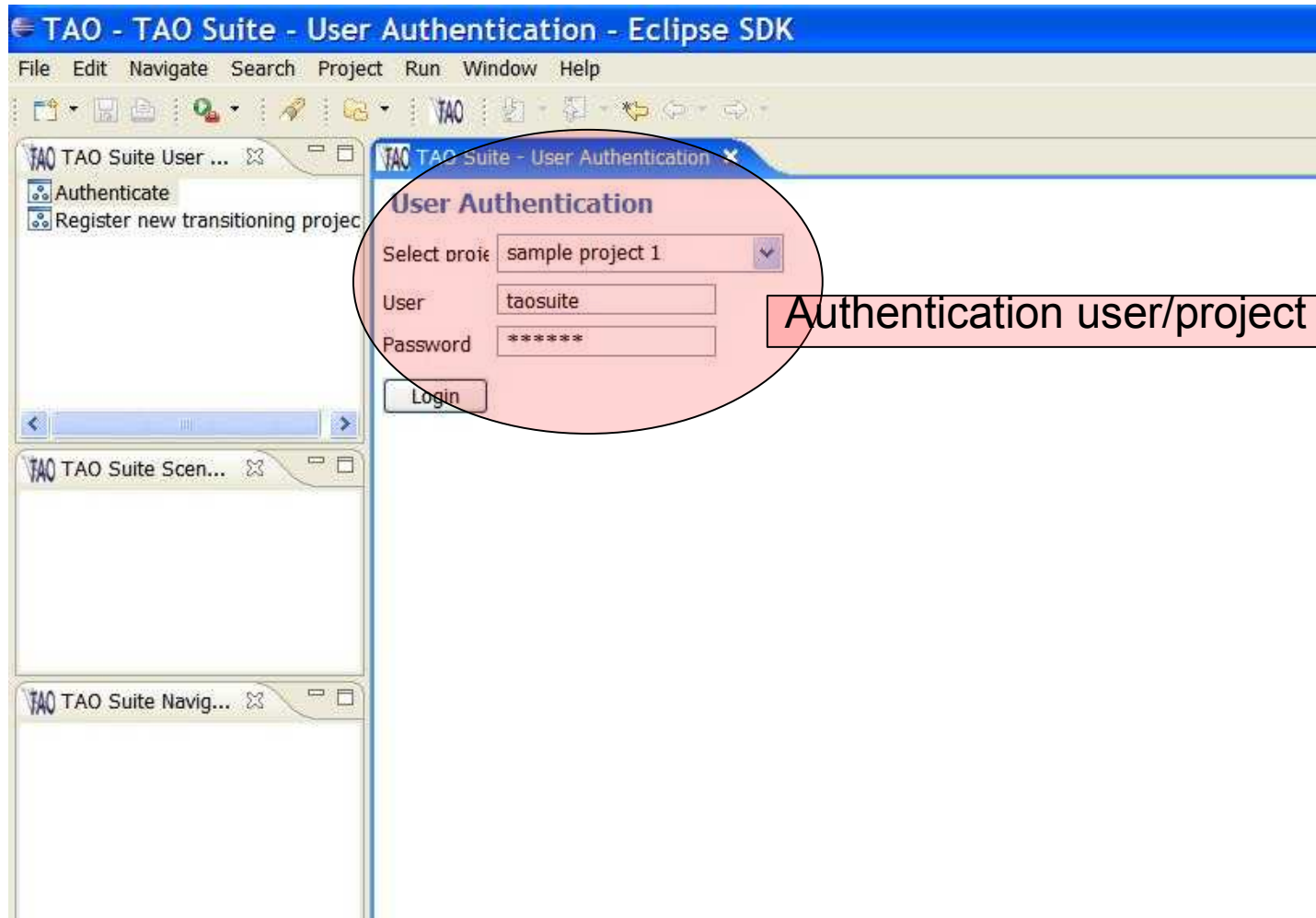


TAO Suite Prototype

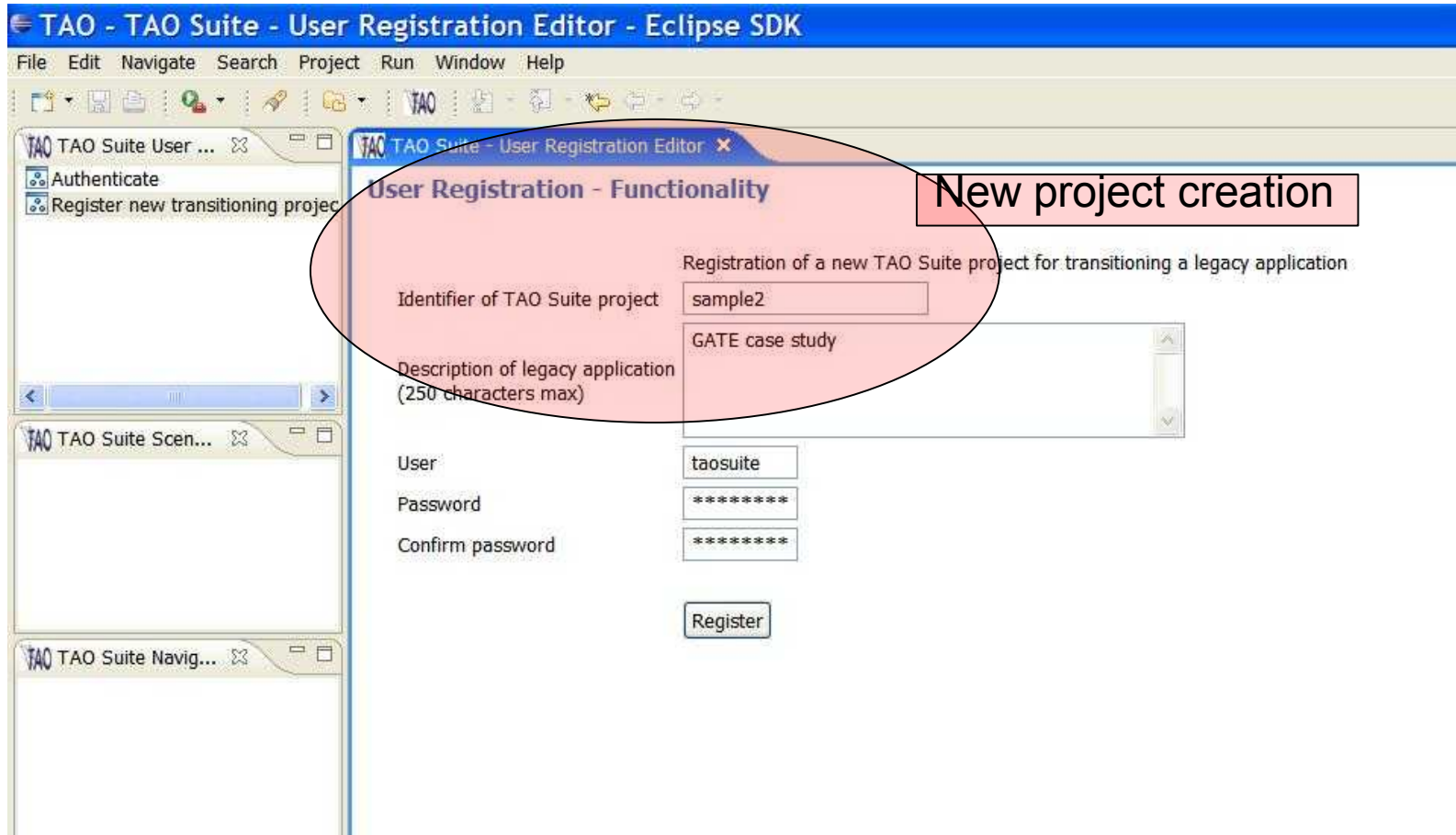
◆ Demo of TAO Suite pre-prototype



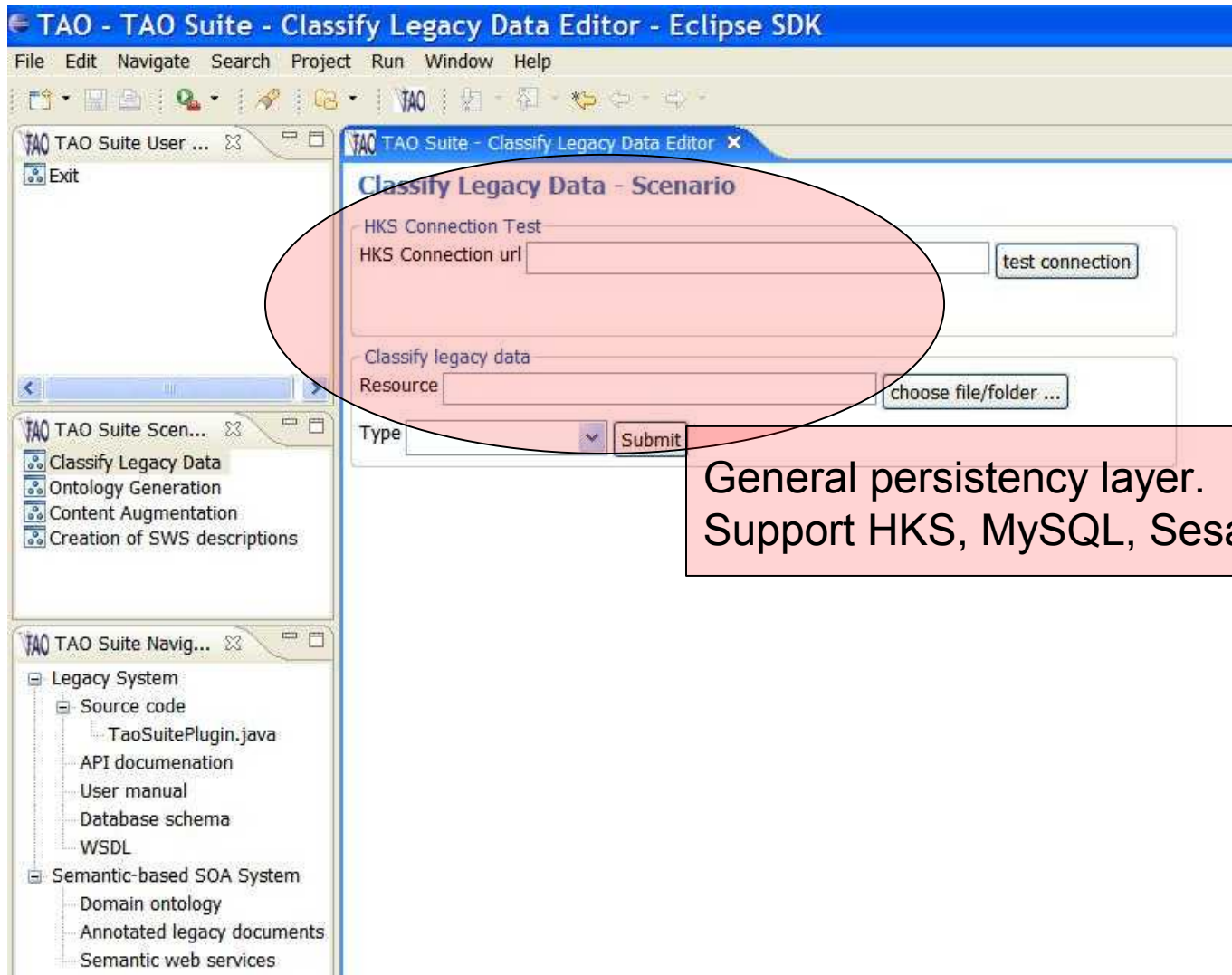
TAO Suite - features



TAO Suite - features



TAO Suite - features



TAO Suite - features

The screenshot displays the TAO Suite Eclipse SDK interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The 'External Tools' menu is open, showing options for '1 OntoGen', 'Run As', 'External Tools...', and 'Organize Favorites...'. A callout box highlights this menu with the text 'Quick access to external tools'. The main workspace shows the 'Ontology Generator' configuration page, which includes sections for 'Latino WS Connection test', 'Latino adapters', 'Create document network', and 'Store refined ontology'. The 'Latino adapters' section lists 'Existing adapters' with 'LatinoGateDemo.jar' and a 'Remove adapter' button. A callout box highlights this section with the text 'Runtime hot deployment'. The 'Create document network' section includes fields for 'Source element', 'Selected adapter', '.doc file destination', and '.bow file destination', each with an 'Attach selected' or 'Browse' button. The 'Store refined ontology' section includes a 'Refined ontology' field and a 'Store refined ontology' button. The left sidebar shows a project tree with folders like 'Legacy System', 'Source code', 'API documentation', 'User manual', 'Database schema', 'WSDL', 'Semantic-based SOA System', 'Domain ontology', 'Annotated legacy documents', and 'Semantic web services'. The bottom left corner of the slide contains the number '34'.

Quick access to external tools

Runtime hot deployment

TAO Suite - features

TAO - TAO Suite - Content Augmentation Editor - Eclipse SDK

File Edit Navigate Search Project Run Window Help

TAO Suite User ...

Exit

TAO Suite - Content Augmentation Editor

Content Augmentation - Scenario

CA Manager connection test

CA Manager url:

Response:

Content Augmentation Manager Adapters

Available adapters	Adapter content
GATE_and_KnowledgeStore	
Test	
GATE_Metadata	
GATE	

Content Augmentation Operations

Content Augmentat

Ontoloov Populat

Adapters for CA Manager

TAO Suite - features

TAO - TAO Suite - Creation of SWS descriptions Editor - Eclipse SDK

File Edit Navigate Search Project Run Window Help

TAO TAO Suite User ...

Exit

TAO TAO Suite - Creation of SWS descriptions Editor

Creation of SWS descriptions - Scenario

CA MANAGER URL

List of CA MANAGER adapters available in CA MANAGER Server for creation of SWS descriptions

- GATE_SWS.xml
- DASSAULT_SWS.xml
- test_SWS.xml

Select the Web Services in "TAO Suite Navigator" view for creating the SWS descriptions

Different adapters for creation of SWS

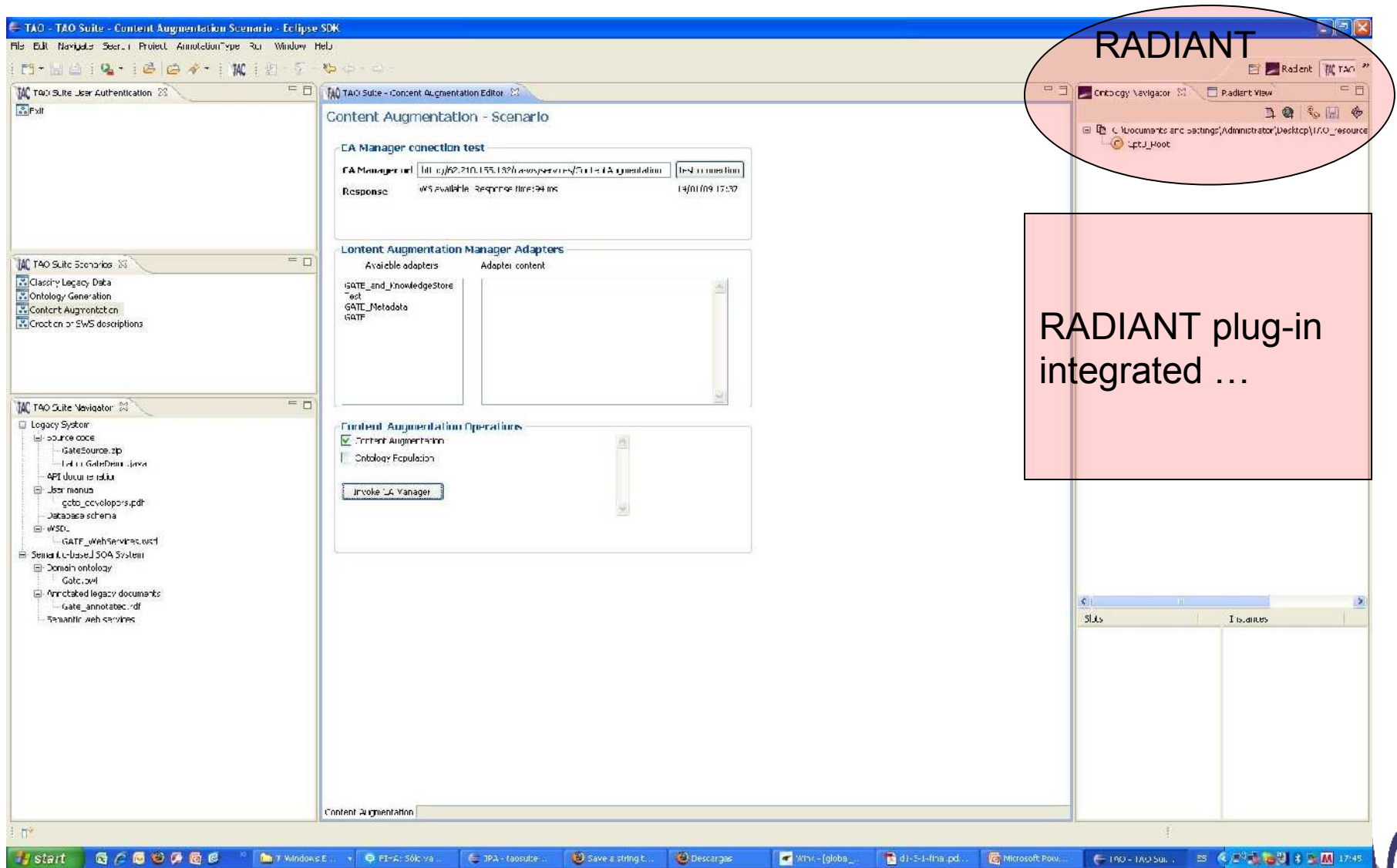
TAO TAO Suite Scen...

- Classify Legacy Data
- Ontology Generation
- Content Augmentation
- Creation of SWS descriptions

TAO TAO Suite Navig...

- Legacy System
 - Source code
 - TaoSuitePlugin.java
 - API documentation
 - User manual
 - Database schema
 - WSDL
- Semantic-based SOA System
 - Domain ontology
 - Annotated legacy documents
 - Semantic web services

TAO Suite - features

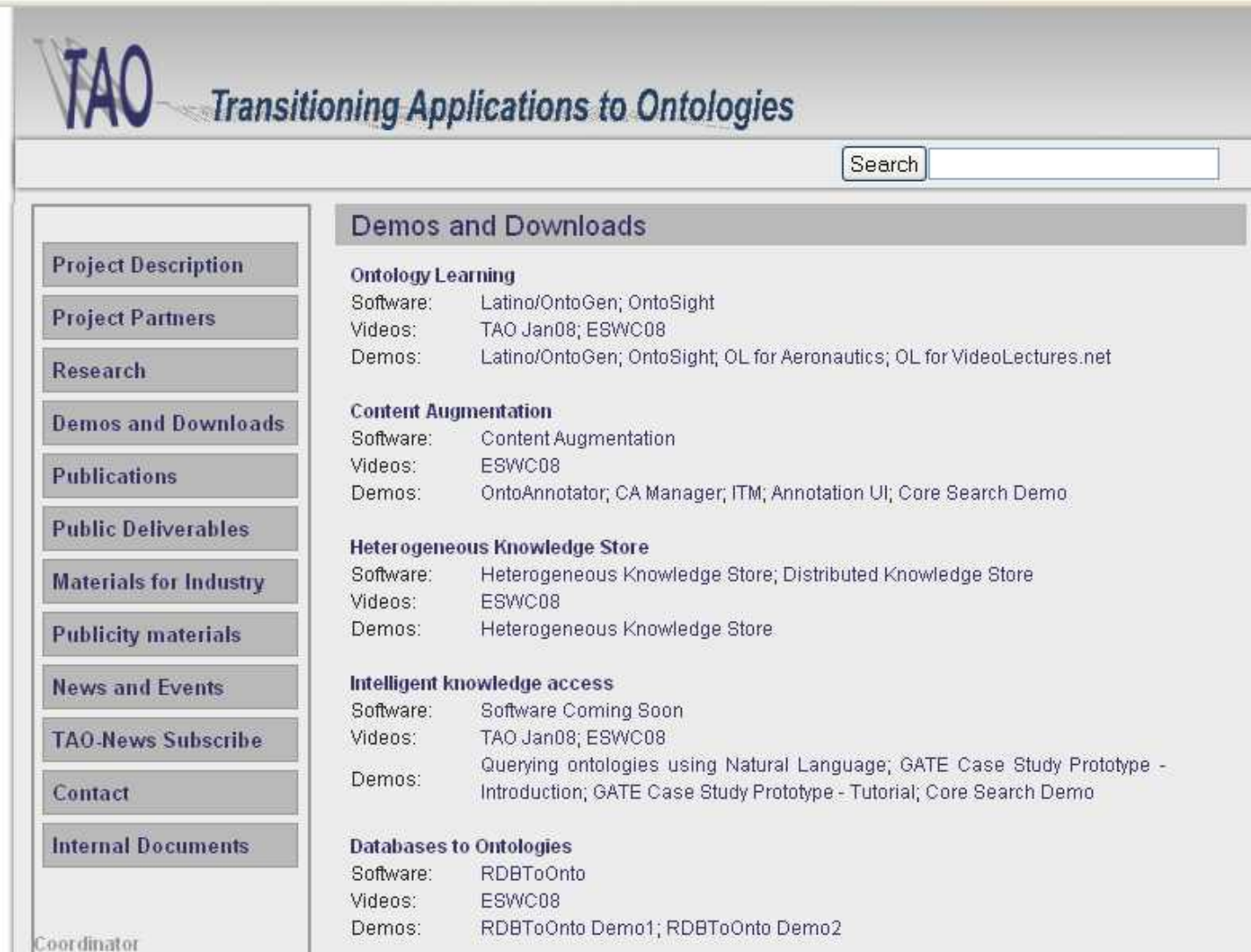


Annotation User Interface

◆ Demo of Annot UI Prototype



TAO Suite – Downloads



The screenshot shows the TAO Suite website interface. At the top left is the TAO logo and the text "Transitioning Applications to Ontologies". To the right is a search bar with the word "Search" and an input field. Below the header is a vertical navigation menu on the left with buttons for: Project Description, Project Partners, Research, Demos and Downloads (highlighted), Publications, Public Deliverables, Materials for Industry, Publicity materials, News and Events, TAO-News Subscribe, Contact, and Internal Documents. The main content area is titled "Demos and Downloads" and contains several sections:

- 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

At the bottom left of the page, the text "Coordinator" is visible.



TAO Suite – Downloads

Coordinator

Kalina Bontcheva
University of Sheffield UK

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
SWESE'08 Workshop

"Call 3 in Motion" workshop at EC

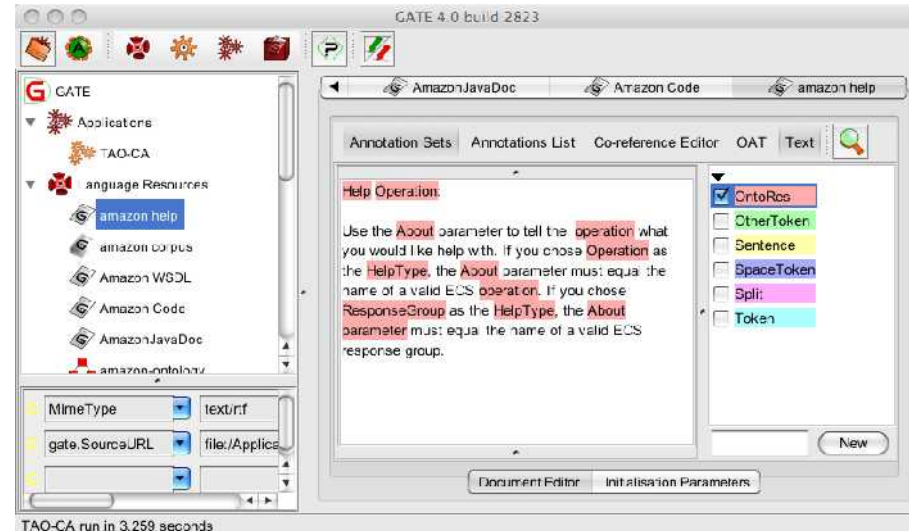
TAO demos
TAO Publicity Materials
TAO Public Deliverables
TAO Project presentation
TAO Project Demo

End

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

Amazon Demo

◆ Demonstration of the Methodology using the Amazon Services

◆ Illustrates:

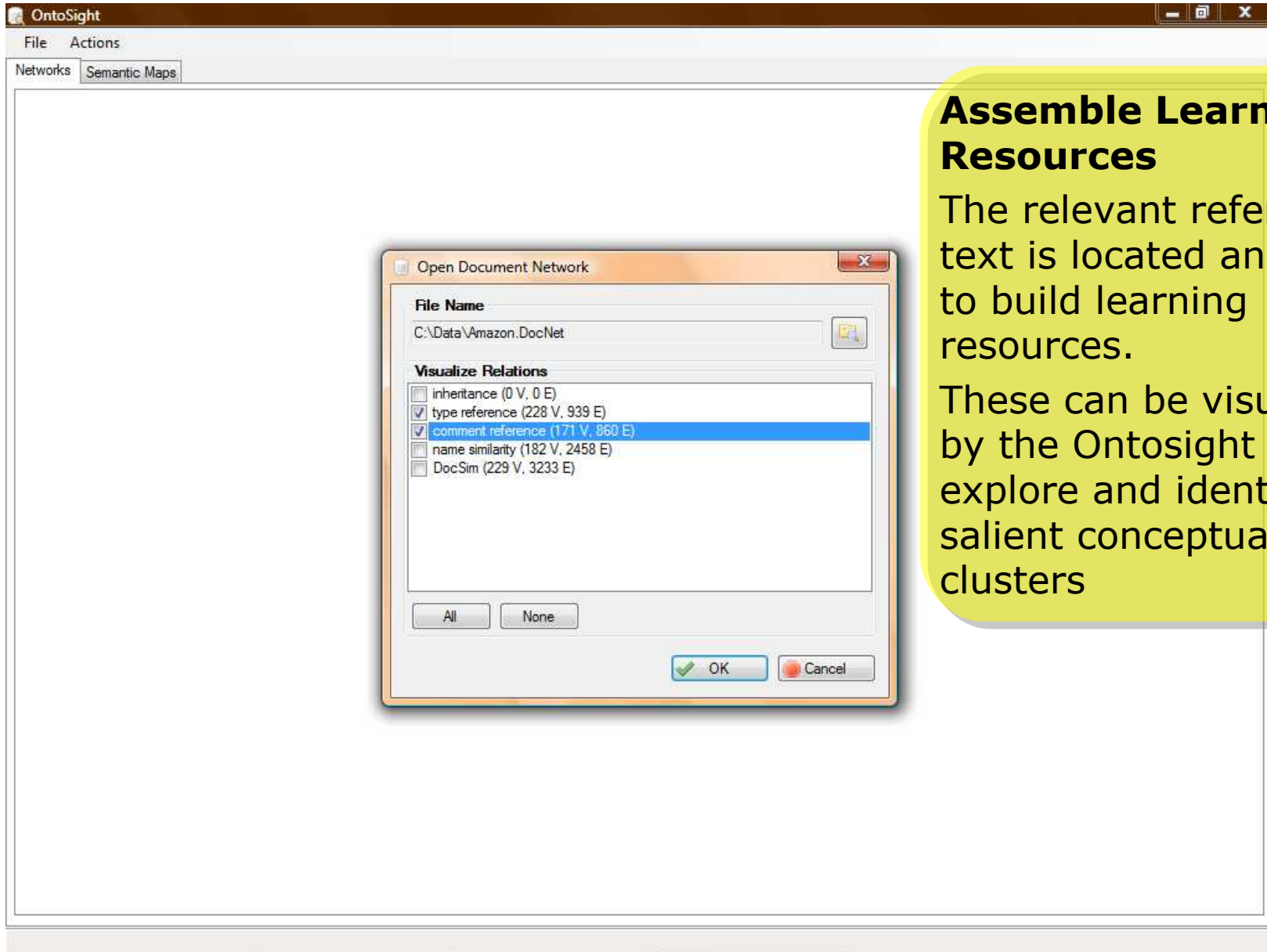
◆ **The ontology learning process** using OntoSight and OntoGen over Amazon data (*screenshots*)

◆ **The annotation process** using Gate (*video*)

The image displays several screenshots from software tools used in the methodology:

- OntoSight:** A 3D visualization of an ontology, showing a central node with many surrounding nodes connected by edges, rendered in a blue and white color scheme.
- OntoGen:** A hierarchical tree structure of concepts, with a root node and several levels of sub-nodes.
- GATE:** The General Architecture for Text Engineering interface, showing a list of applications and resources, including Amazon services like Amazon S3, Amazon EC2, and Amazon IAM.
- Document Editor:** A window showing a help page for the 'About' parameter of an operation, with a list of tokens and a 'New' button.

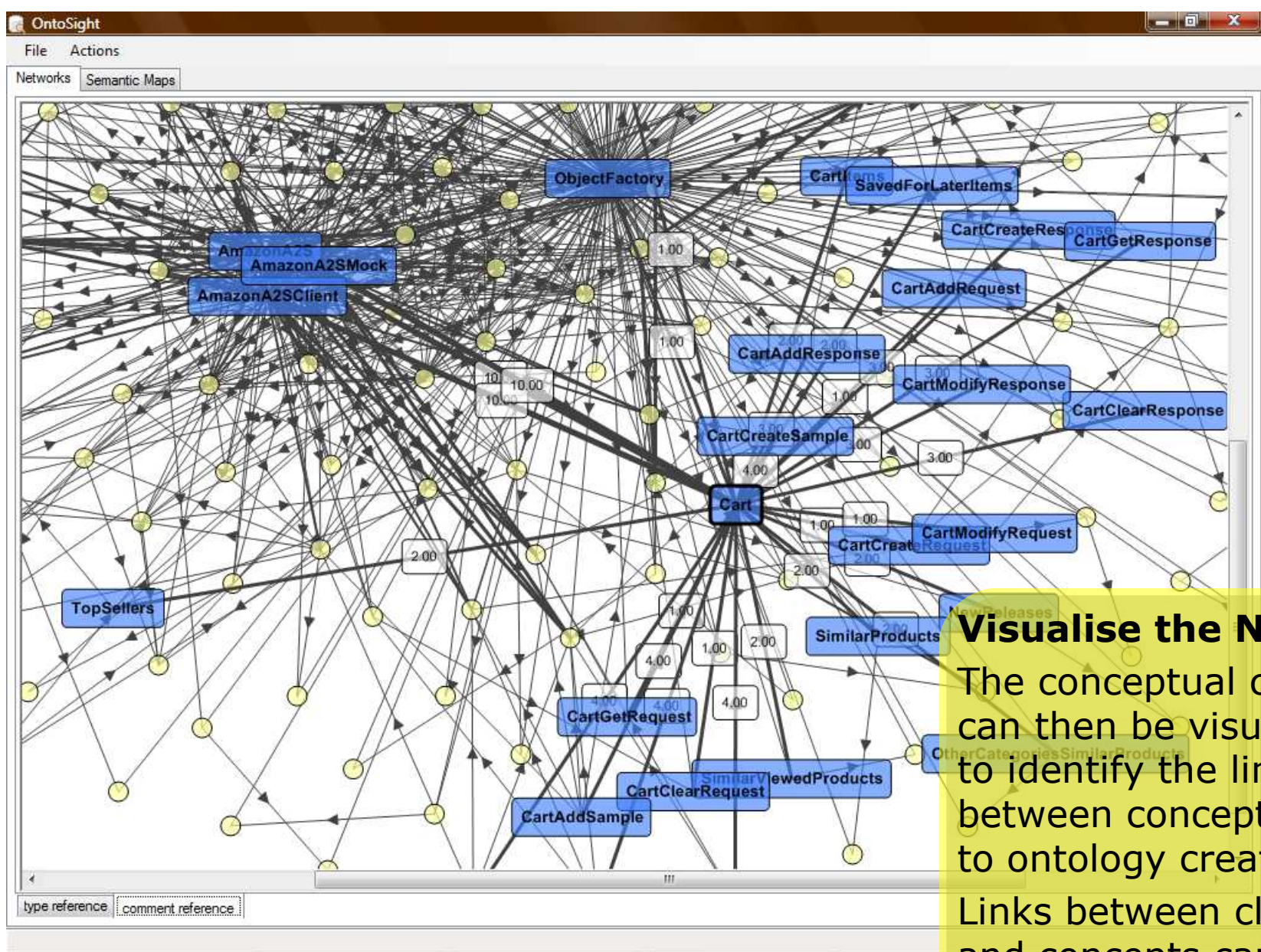
TAO CA runs in 3.286 seconds



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

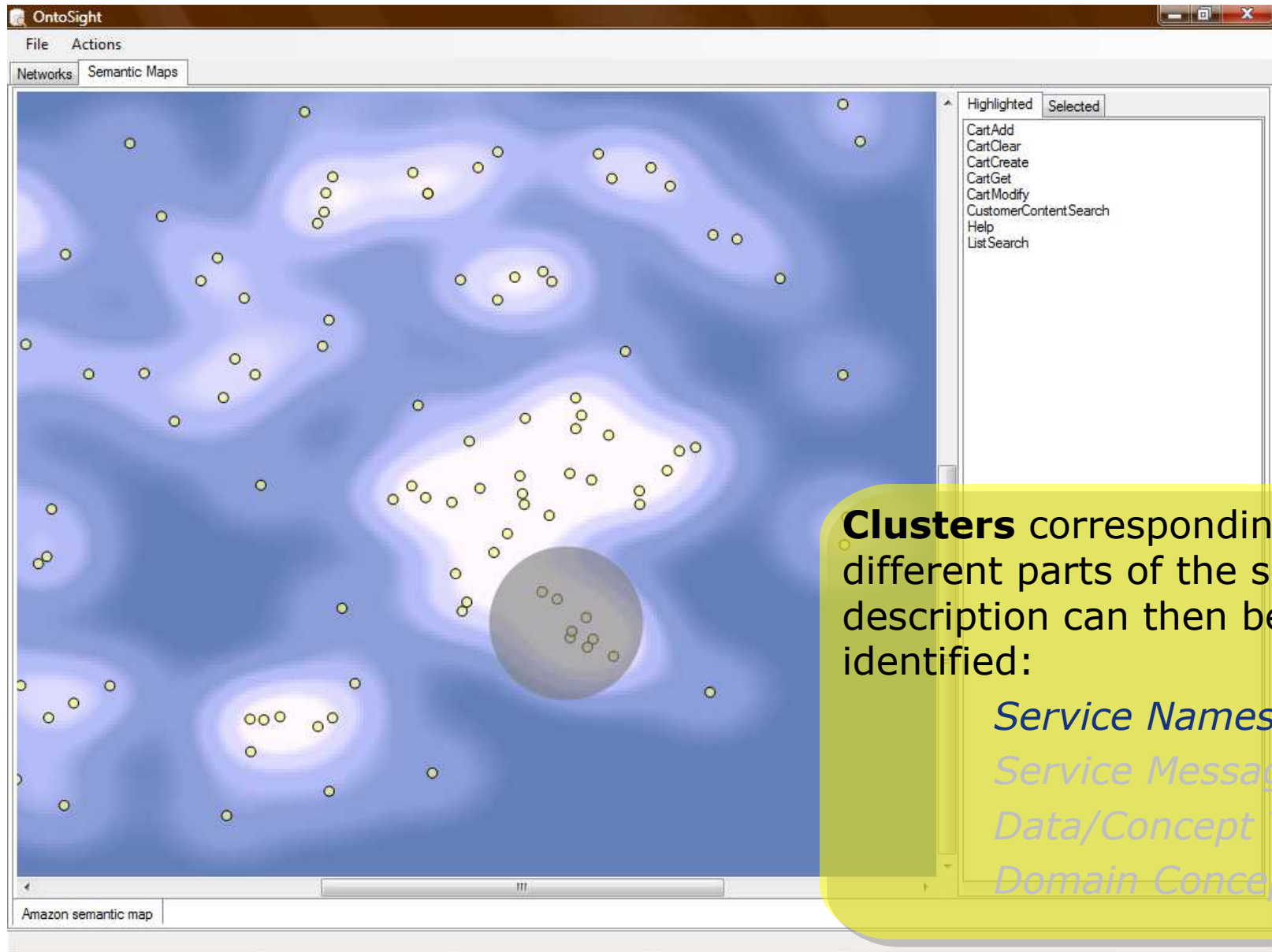


Visualise the Network

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.

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:

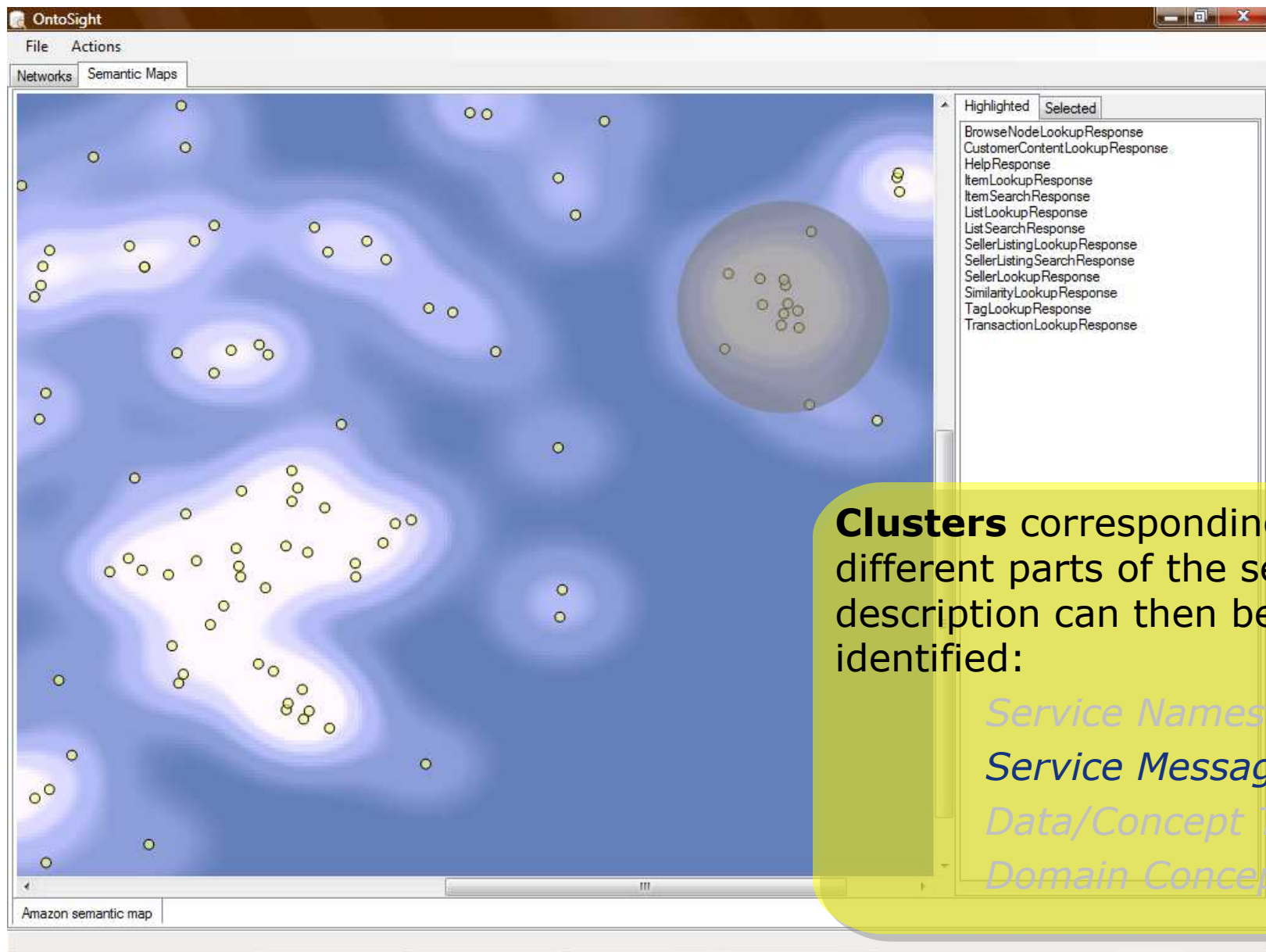
Service Names

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

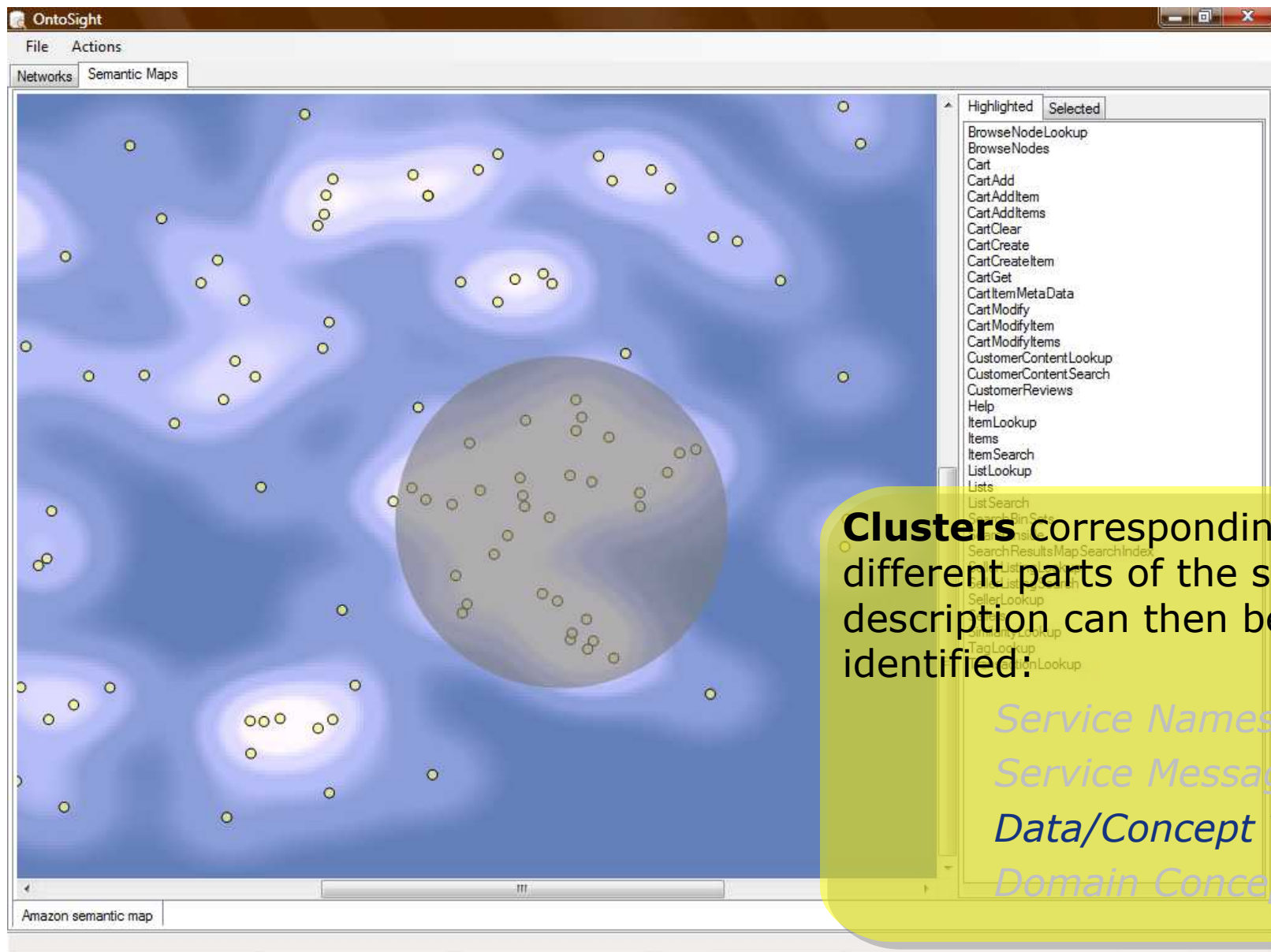
Service Messages

Data/Concept Types

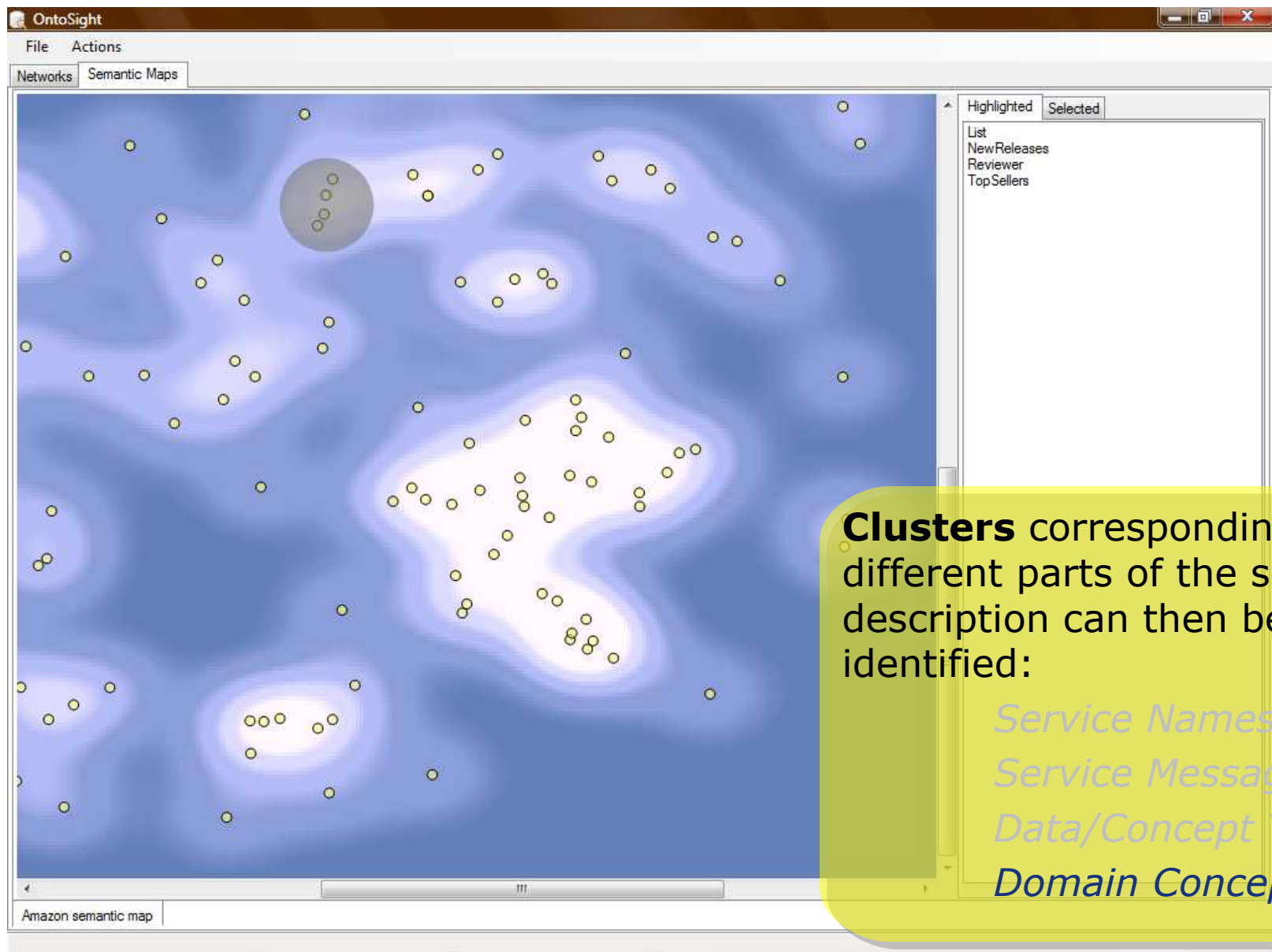
Domain Concepts



The visualization of a semantic space and the use of the context tool for exploring the space



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

The screenshot displays the OntoGen for Tao software interface. On the left, a 'Concepts' tree lists various concepts like 'Subjects, Accessories, Tags' and 'CartCreate, CartGet, CartModify'. The main area shows an 'Ontology details' window with 'Ontology visualization' selected. A central 'root' node is surrounded by numerous concept clusters, each in a blue box. A yellow callout box with the text 'Build Ontology Based on the identified clusters, a taxonomic ontology can be generated as an OWL file' is overlaid on the right side of the visualization. The 'Concept properties' panel at the bottom left shows 'Suggestions' and 'Relations' tabs, with 'k-Means' selected and 'Suggest' button visible.

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

Concepts
New Move Delete
root
Subjects, Accessories, Tags
CartCreate, CartGet, CartModify
AmazonA2SConfig, AmazonA2SLoc
Review, Reviewer, Feedback
SimilarityLookupResponse, CartGetF
HTTPHeaders, Header
Language
ImageSets, Image, EditorialReviews
ListSearchSample, ListLookupSamp
DefaultResponseGroups, AvailableF
ItemLookup, SimilarityLookup, ListLc
CollectionItem, CollectionParent, Oth
SellerLookupRequest, SellerListingL
AmazonA2SMock, AmazonA2SClier
Arguments, OperationRequest, Argu
CartCreateRequest, CartModifiReq

Ontology details
Ontology visualization | Concept's documents | Concept Visualization
Concept font size: 11 | Show relation type | Relation font size: 9

Concept clusters (examples):
- NonNegativeInteger, ItemLookup, SellerLookupRequest, SellerListingLookupRequest, SimilarityLookupRequest, ListLookupRequest
- CustomerContentLookupResponse, Guide, Tagging, CustomerContentLookupSample, CustomerContentLookupRequest, CustomerLocation
- TransactionItems, ShipmentItems, ChildTransactionItems
- VariationSummary, CollectionSummary, OfferSummary
- TopSeller, SellerListing, Seller
- Subjects, Accessories, Tags
- AmazonA2SConfig, AmazonA2SLocale, AmazonA2SException
- Review, Reviewer, Feedback
- SimilarityLookupResponse, CartGetResponse, CartGetRequest
- HTTPHeaders, Header
- Language
- EditorialReviews
- ImageSets, Image, EditorialReviews
- ListLookupSample, CustomerContentSearchSample
- DefaultResponseGroups, AvailableResponseGroups, AvailableParameters
- ItemLookup, SimilarityLookup, ListLookup
- CollectionItem, CollectionParent, OtherCategoriesSimilarProduct
- SellerLookupRequest, SellerListingLookupRequest, TransactionLookupRequest
- AmazonA2SMock, AmazonA2SClient, AmazonA2S
- Arguments, OperationRequest, Argument

Concept properties
Details | Suggestions | Relations
k-Means | Suggest | Query | Add
No. suggestions Docs: All (selected) | Unused
Keywords | No. ... | [%]



Amazon ECS Ontology

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

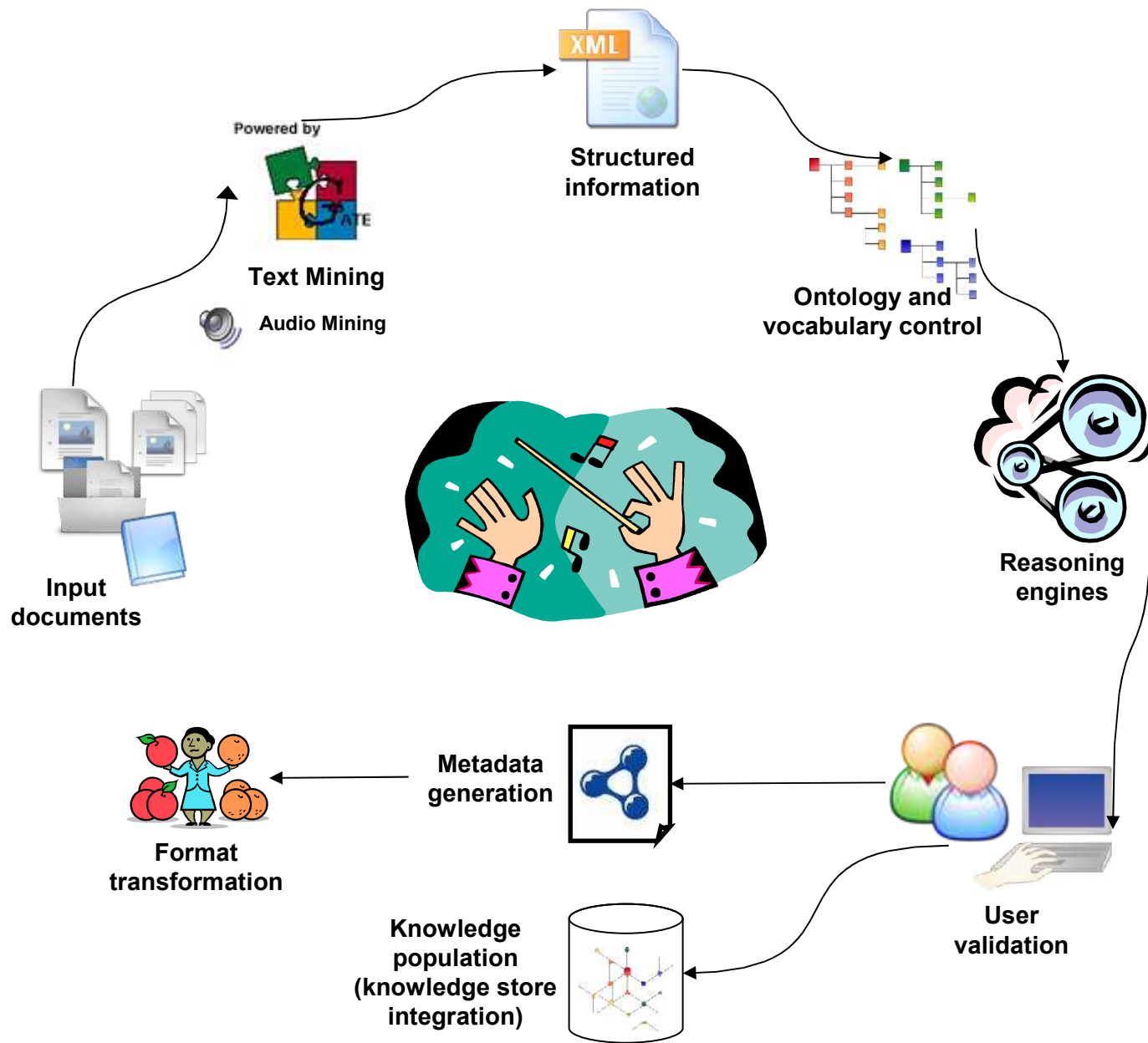
The screenshot displays the GATE 4.0 build 2823 interface. The main window shows a document editor with the text "Help Operation: Use the About parameter to tell the operation what you would like help with. If you chose Operation as the HelpType, the About parameter must equal the name of a valid ECS operation. If you chose ResponseGroup as the HelpType, the About parameter must equal the name of a valid ECS response group." The text is annotated with red boxes. The left sidebar shows the GATE application tree with "Language Resources" expanded, listing "amazon help", "amazon corpus", "Amazon WSDL", "Amazon Code", "AmazonJavaDoc", and "amazon-ontology". The right sidebar shows the "Annotation Sets" panel with "OntoRes" selected. The bottom status bar indicates "TAO-CA run in 3.259 seconds".

End

Questions? Discussion welcome!

Achieved in second year : CA-Manager specifications

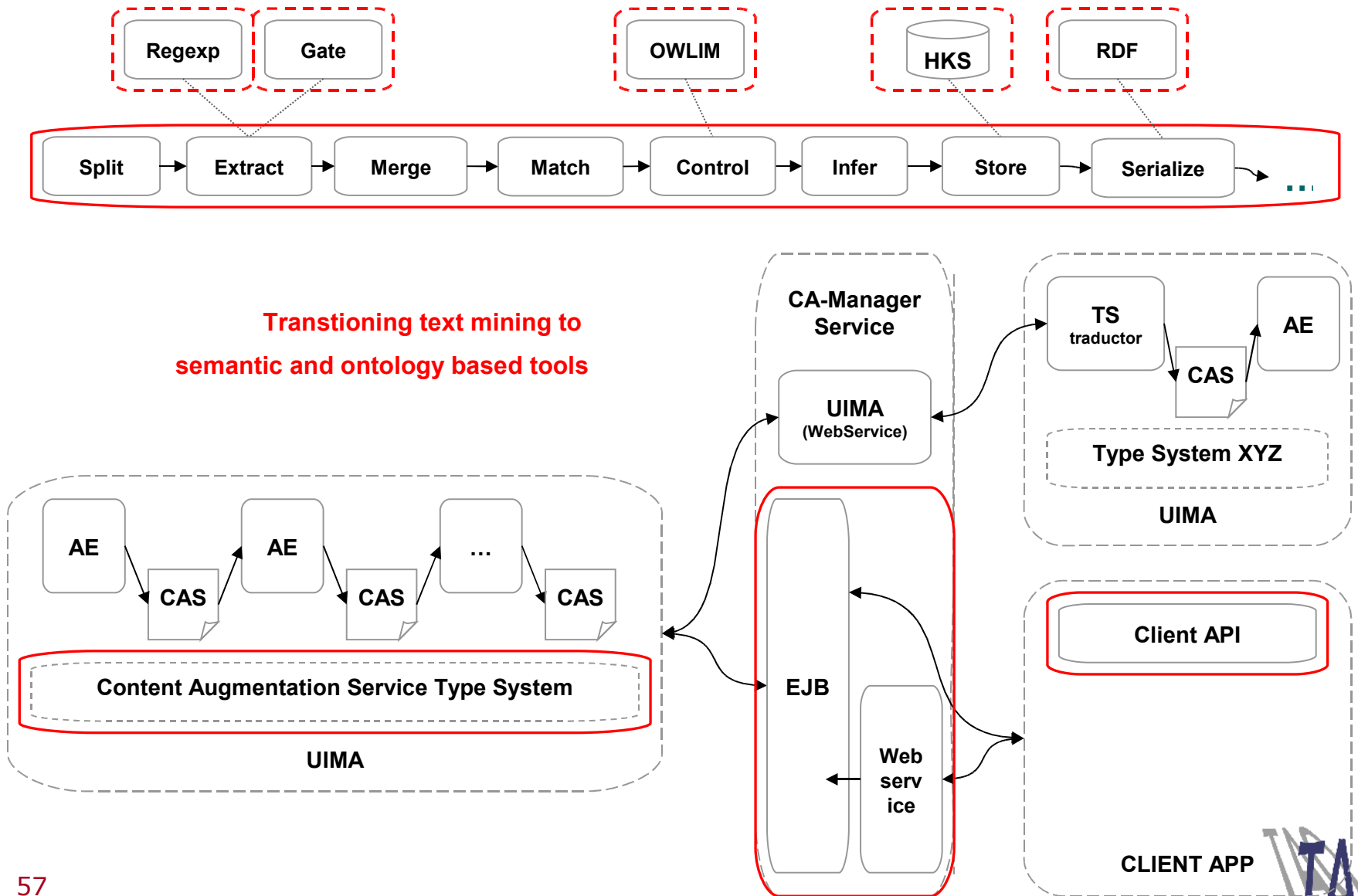
- ◆ CA-Manager : middleware used by TAO suite that connects text-mining, knowledge repository, + other components if needed
- ◆ Must be able to address :
 - ◆ Automatic **metadata generation**
 - ◆ Generate a set of metadata about the processed document
 - ◆ Ex : create a searchable database of document metadata (lifescience 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



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



Achieved in second year : CA-Manager architecture

- ◆ The architecture
 - ◆ Is a specialized UIMA pipeline
 - ◆ Focuses on transioning 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