# Authoring resources in a concept-based educational system

Rijsdijk, Jurgen T.J.

*Award date:*
2006

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

MASTER'S THESIS

# Authoring resources in a concept-based educational system

by
J.T.J. Rijsdijk

Supervisor: Dr. L.M. Aroyo

Eindhoven, January 2006

# I. Preface

This document is the result of my graduation project. The graduation project concludes the Technical Computer Science course (Technische Informatica) at the Technische Universiteit Eindhoven (TU/e).
The graduation project was carried out at the Information Systems (IS) group of the Faculty of Mathematics and Computer Science (Faculteit Wiskunde en Informatica) at the TU/e.

I would like to thank the following persons for their contribution, ideas for improvement, discussions and support to the project:

Dr. Lora Aroyo
Prof. Dr. Paul De Bra
Dr. Ad Aerts
Ir. Johan Hubens
Ir. Bas Rutten
Ir. Kees van der Sluijs

Finally I would like to thank the fellow graduation students of rooms 7.45 / 5.41 and classmates for ideas and support.

Eindhoven, January 2006.
Jurgen Rijsdijk.

# II. Table of contents

# III. List of figures

# IV. Definitions

| Word | Meaning |
|---|---|
| Library | 1. Set of resources, resource pool<br>2. Object providing functionality in Java<br>3. Collection of searchable data |
| Metadata | Data about data, e.g.: information about a resource |
| Resource | Object that can be described with metadata, e.g. a book / a webpage / a person / a learning system / etc. |
| Concept | Name defining an object of a knowledge domain. |
| Cluster | Grouping structure, set of resources sharing a commonality |
| Domain | Knowledge domain, collection of concepts |

# V. Acronyms

| Acronym | Full name |
|---|---|
| AIMS | Agent based Information Management System |
| API | Application Program Interface |
| CST | Course Sequencing Tool, OntoAIMS module |
| DAML | DARPA Agent Mark-up Language |
| DAML+OIL | DAML with Ontology Inference Layer, metadata specification |
| DC | Dublin Core, metadata specification |
| DM | Data model |
| DMM/DMT | Domain Management Tool (or Module), OntoAIMS module |
| HTML | HyperText Mark-up Language |
| HTTP | HyperText Transfer Protocol |
| IEEE | Institute for Electrical and Electronics Engineers |
| LMS | Learning Management System |
| LOM | Learning Object Metadata, metadata specification |
| LTSC | Learning Technology Standards Committee |
| NS | Name Space, XML definition |
| OntoAIMS | Ontology version of AIMS |
| OWL | Ontology Web Language, metadata specification |
| P&F | Presentation & Feedback, RMT component |
| RDF | Resource Description Framework, metadata specification |
| RDFS | RDF schema, metadata specification |
| RMT | Resource Management Tool, OntoAIMS module |
| SCORM | Sharable Content Object Reference Model, ADL standard |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator, part of URI |
| URN | Uniform Resource Name, part of URI |
| XML | eXtensible Mark-up Language |
| W3C | World Wide Web Consortium |
| WBIS | Zie WIS |
| WIS | Web-based Information System |
| WWW | World Wide Web |

# 1. Introduction

## 1.1 Web-based information systems

Web Information Systems or Web Based Information Systems (WIS) use the open data structure of the Web and Web technologies to gather/retrieve information (data) from sources connected to the Web and present the information in a web (hyper-) media presentation to the user.

These WIS address various application domains such as e-business, education, geography, etc. As any other Information Systems, WIS are designed to manage very large sets of data and to offer specialized services.

An important task for WIS is to bridge the gap between a collection of heterogeneous and dynamic data sources and a group of users with different preferences using different platforms for accessing the information. The diversity of Web users makes a one-size-fits-all approach not applicable.

The appropriateness of information a WIS delivers to its users turns out to be an acute problem when, designing such systems, one ignores that

1) All users do not need the same information, and
2) Users do not need all the available information all the time.

## 1.2 Open standards for metadata interchange

Within the scope of authoring educational web-based information systems (WBIS / WIS), there is a substantial research effort towards semantic integration of educational resources based on using metadata annotation and metadata representation.
The following specifications emerge as standards for representing semantics of data (metadata) in information systems both web-based and stand-alone:

- Dublin Core (DC, [DC])
- Learning Objects Metadata (LOM, [LOM])
- Resource Description Framework (RDF, [RDF])
- RDF Schema (RDFS, [RDFS])
- DARPA Agent Mark-up Language (DAML, [DAML])
- DAML + Ontology Inference Layer (DAML+ OIL, [DAMLOIL]) and
- Web Ontology Language (OWL, [OWL])

Among the different aspects of DC, LOM, RDF (S), DAML+OIL and OWL the encoding of metadata appears useful for educational applications, since it facilitates further reusability, easier manipulation, update and maintenance of information resources within educational web-based systems

## *1.3 initial situation: AIMS*

Previous research on online course support has been performed on the concept-based educational system AIMS. The AIMS system is intended to support both students (the users / viewers of educational content) and teachers (the authors of the educational content). For this reason AIMS constitutes of two main parts – a student (viewer) environment and a teacher (author) environment. Figure 1 shows the division of the two environments and the structure connecting them.

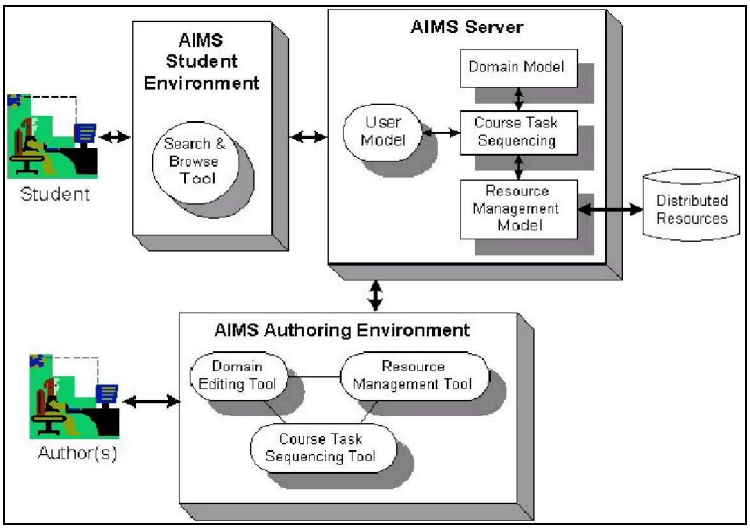


**Figure 1, AIMS architecture [Aroyo2]**

Figure 1 shows the components of the AIMS system
The system is composed of the student (viewer) environment, the authoring (teacher) environment and the server.
The server connects the user environments and contains the available metadata.
The resource management model is a small set of metadata describing resources and linking[1] to the described resources at different locations.

### 1.3.1 Teacher component

The AIMS teacher component provides the means to edit domain, the library (resources) for the domain and the course for the domain.
The authoring environment provides three tools for the process of course content management:

- Domain editor
- Resource editor
- Course editor

[1] Linking is only done if the resource refers to an external URL.

The domain structures can be managed using the Domain editor. A domain model is composed of domain concepts, relations between concepts and relations of concepts with resources from the library
For a domain the following parts can be edited:

- Domain description (domain name, author, description, dates, etc.)
- Concepts of this domain and links to other concepts
- Types[2] of links between concepts
- Links to library resources

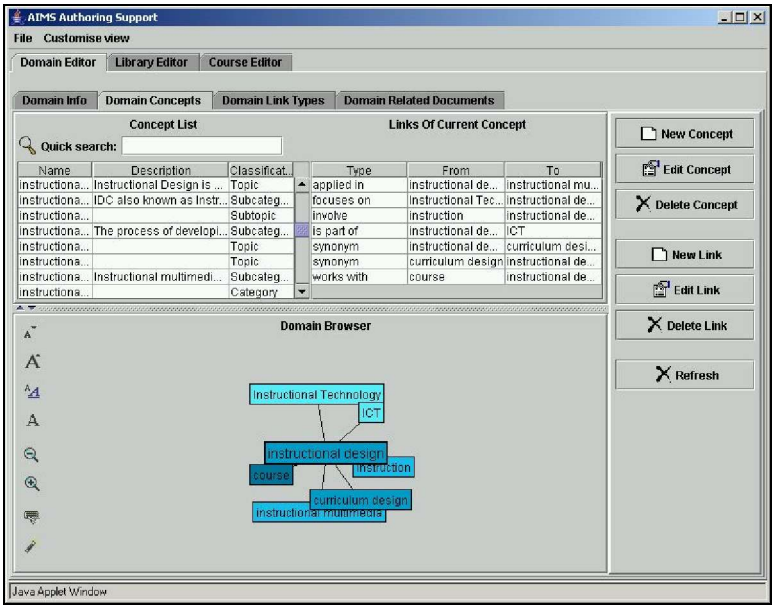


**Figure 2, screenshot AIMS domain authoring [AIMS]**

Figure 2 shows the AIMS user interface where the author can edit AIMS data. Active in this view is the part where the author edits domain data and specifically the part where the author can manage domain concepts and can manage the links between concepts.

[2] The types of links (relationships) between concepts are labels defined by the author. Example types are: 'applied in', 'compared to', 'has a', 'influences', etc.

The resource library is managed using the Resource editor. The library contains metadata about educational resources that can be linked to concepts in the domain model and to each other.
For a library of the domain the following parts can be edited:

- Library description (Library name, author, description, dates, etc.)
- Library documents (metadata sets)

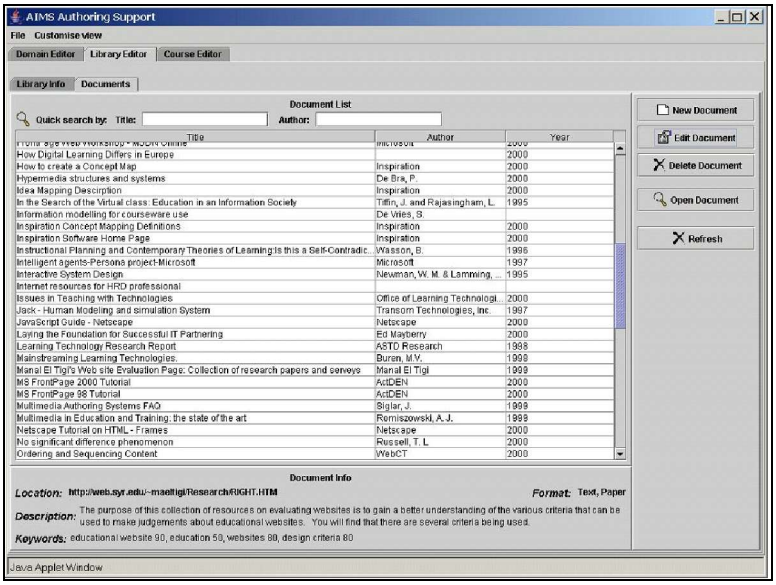


**Figure 3, screenshot AIMS library authoring [AIMS]**

Figure 3 shows the AIMS user interface where the author can edit AIMS data. Active in this view is the part where the author edits library data and specifically the part where the author can manage documents (resources).

Using the Course editor the teacher can build courses. The teacher defines course tasks and he selects a set of domain concepts to be studied by the students for each task. The teacher can browse through the set of resources related to those concepts in order to select the ones treated in the course. The set of resources is generated by AIMS automatically on the basis of the links between domain concepts and educational resources.

### 1.3.2 Student component

The AIMS student component is a concept-based information viewer that lets the viewer browse through information, relations and course structures. The student taking the course evaluates the elements of the course and decides what to read up on, what links and relations to follow and what parts to discard (not interested or already known). A user model tracks the learning advances of the student to help the student pick interesting or required concepts to complete the course task.

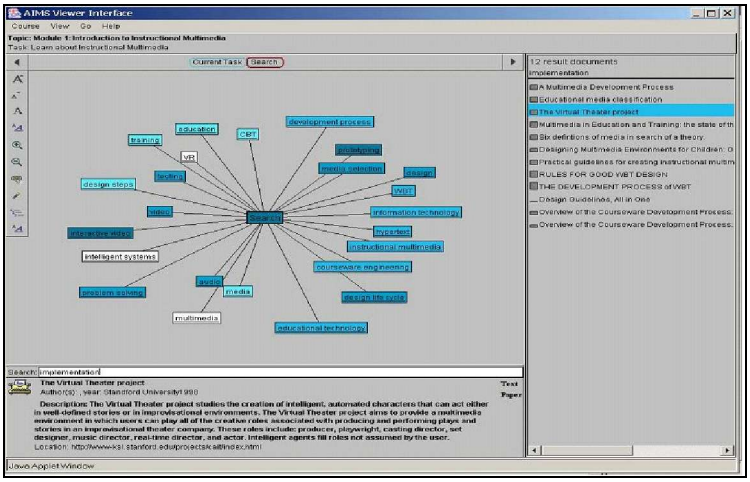


**Figure 4, screenshot AIMS student environment [AIMS]**

Figure 4 shows the student viewer screen for browsing a course task in the AIMS student environment and the related concepts and resources. The student starts with a course task and browses relations to find out about concepts, relations and read resources.

## 1.4 From AIMS to OntoAIMS

This project continues some of the research done on the AIMS system. The project will focus on creating a successor to the library part of the existing AIMS teacher (author) environment by implementing means to efficiently manage and search through the resource pool / library. In this project open standards for educational metadata will be used to exchange data with other systems in a standardised way.

Important here will be the communication and sharing of information with the course-construction environment and with the domain-authoring environment. This aids in the semantic integration of the educational resources. The resources are of diverse nature and are reusable across domains and courses. Therefore the presentation and manipulation of resources needs to be domain and course independent. The desired data dependencies between the OntoAIMS modules are visualised in Figure 5

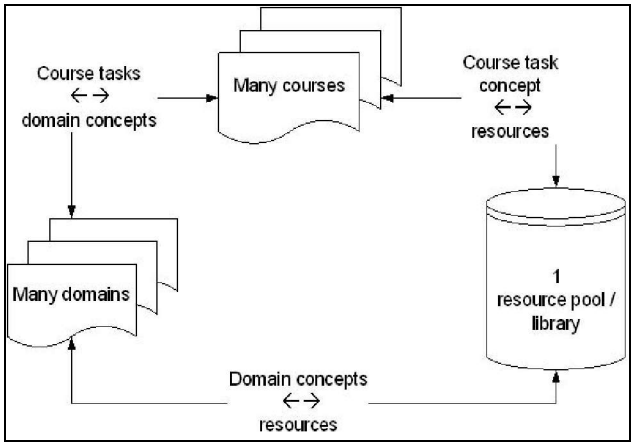


**Figure 5, OntoAIMS module relations**

In the current AIMS implementation it requires a lot of time and effort from the instructor to create a course. The author needs to have a detailed domain model, resource annotations, and links between the concepts in the domain model and the resources. Everything has to be built from scratch, since there is no possibility to import external information, and no open standards are supported (AIMS stores its information in proprietary formats). In AIMS the system keeps its sets of resources and domain for a course together as a set. No information is reusable between AIMS sets. The strategy for OntoAIMS is to keep resources separate from the course structure and only refer to resources in the resource pool when building a course or connecting concepts. This way all resources can be referred to from all concepts and all courses enabling better reuse.

The structure of the data in AIMS is as shown in Figure 6

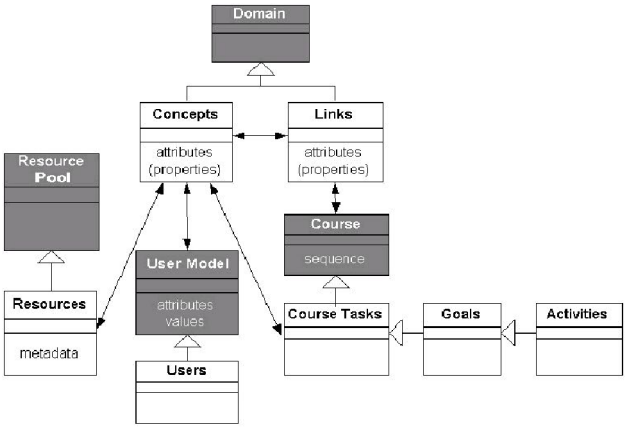


**Figure 6, AIMS information model [Aroyo2]**

Figure 6 shows the information structure for AIMS data. AIMS data sets are composed of one Domain with one Resource Pool, one User Model and a collection of Courses. Concepts in the Domain are connected with Links defined by the author. Course tasks have (link to) concepts. Resources describe (link to) Concepts.

In OntoAIMS we want to reuse as much as possible and therefore the system is split into three subsystems that each manage specific data objects (courses, concepts and resources) and share information between them. With this separation data objects are reusable across all domains and all courses.

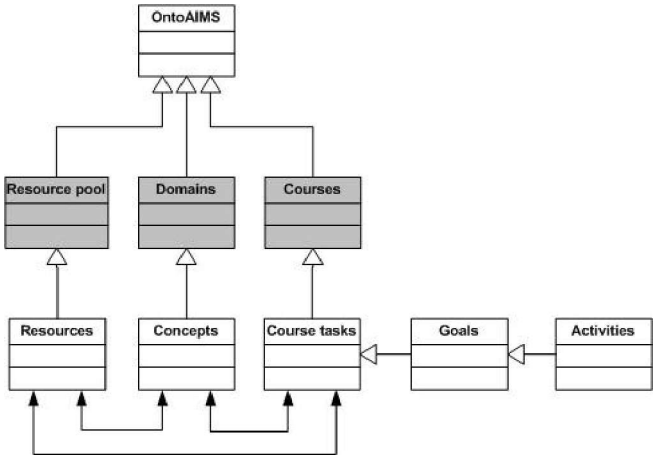


**Figure 7, OntoAIMS information model**

Figure 7 shows the information structure for OntoAIMS data. OntoAIMS has three distinct data sets: a Resource pool, multiple Domains, multiple Courses. Resources describe (link to) Concepts. Course tasks have (link to) Concepts and Resources. Concepts link to Resources for information about the Concept.

In OntoAIMS we use web standards for data storage and data exchange. Resource annotations are defined using an open standard and are represented using RDF [RDF]. Domain models are represented using one of the emerging semantic web standards for knowledge representation: OWL [OWL]. Course sequencings will be SCORM-compliant [SCORM]. The advantage of using web standards is that it gives the opportunity to exchange information with other systems that use such standards without data conversion. This can significantly improve the usability of the system, by speeding up the authoring process. Instead of defining all resource annotations and building all domain models themselves, instructors can use already available resource annotations or general domain ontology's (and alter these to their specific needs).

The open standards for metadata annotation offer more aspects (metadata fields and categories) for the author to describe resources than AIMS offers.

The AIMS system lacks some support to the instructors:
Information on relations between different items, i.e. a concept and a resource, are only available in lists. The author would have more overview when he sees these relations in a graphical form like the AIMS student environment has.

In OntoAIMS there is more automation in the authoring process, and the instructors are provided with information to improve the knowledge in the system. The OntoAIMS modules will provide more views on the data to further explore relations and information.

Another important issue is to find out what specific functionality and data is needed in the resource authoring for OntoAIMS in contrast with general resource / metadata editors

## *1.5 OntoAIMS resource scenarios*

To sketch the resource-authoring environment, that will be the result of the project, three scenarios are presented. The user of the system, the author of educational content, uses the system in several ways to search for, create and modify resources to use for educational goals.

### 1.5.1 Scenario 1: Creating resources

Thomas is a teacher in the department of Physics and he wants to prepare a course on the subject of Thermodynamics. For this course Thomas has already selected a few books out of his personal library. He selected some readers from the department website and he asked 2 colleagues to support the students with questions.

Thomas has to enter all these resources (books, readers & colleagues) and some resources that he finds with Google or the University Library in to the OntoAIMS system. With these resources and the domain ontology that Thomas has built in the OntoAIMS domain editor the course can later be created.

To enter new resources in the OntoAIMS resource editor Thomas starts the OntoAIMS authoring environment and he goes to the resource editing part of the system. In the resource authoring section he selects that he wants to enter a new resource into the system. For the resource he enters into the system he has to have several properties; the system will give Thomas a set of 'input boxes' to describe the resource. Each of these 'input boxes' has a description and an example. Thomas knows that the more information (metadata) about the (learning) resource he enters into the system the more useable the resource becomes. Thomas repeats the procedure of entering metadata for all the resources he wants to add to the system.
Thomas tells the system to save the resources (permanently) and closes the authoring environment.

### 1.5.2 Scenario 2: Looking through resources

Vincent is one of the colleagues of Thomas on the Thermodynamics course. Vincent wants to look up the coverage on the topic of conversion of energy from one form to another.

The OntoAIMS system has resources on several topics (for use in several different courses) like vehicles, molecules, learning systems, operating systems and more. Vincent is not interested in all of these subjects and to reduce the volume of information to look through Vincent enters some search words into the system. The OntoAIMS system interprets the query and gives feedback to the author if necessary (if there is an error in the query or the query yields no results). The user (resource author; Vincent) is shown a formatted result set for the query.

Vincent wants to know about conversion of energy from one form to another.
He enters the query: "*energy form conversion*"
The system extends the query using synonyms and (equivalence) relations from domains.
With the extended query the system finds 15 resources in the resource pool, resources like "Forms of energy", "Energy sources", "The Atoms family", etc.
These found resources are grouped into sets by evaluating commonalities and relationships between the search results.

Vincent now goes through the search results by clicking on resources, viewing information and browsing the relations between resources, groupings and concepts. When he is done viewing resources and relations Vincent closes the authoring environment.

### 1.5.3 Scenario 3: Updating resources

A semester later, Charlie, a colleague of Thomas, wants to teach a similar course on thermodynamics. Several months have passed since Thomas gave the course and some information might not be accurate any more. Books may have been sold from the University Library, making it very hard for students to study the materials for the course. Moreover information on websites may be moved, deleted or expanded.

Charlie opens the OntoAIMS authoring environment and finds out which resources are used in the course. Using the resource authoring section Charlie looks up the books he needs and cross-references them with the University Library. Charlie deletes resources on books that are no longer in circulation and he adds the new books on relevant subjects that the library has bought. When Charlie deletes the old resources the system tells Charlie that those resources where attached to courses and domains, Charlie already knew this. He will link the new books to those concepts and courses later on.

For web pages that changed or documents that were moved Charlie opens each resource (metadata), one at a time. He instructs the system that he wants to edit the resource. The system shows him the resource metadata in an edit dialog. Charlie changes the information to reflect the move or modifies some properties (description / title / etc.) and selects to save the resource.

After modifying the last resource Charlie tells the system to save the resources (to disk) and closes the authoring environment.

# 2. Project definition

Part of the project will be to explore and analyze current standards on representing metadata. And evaluate existing tools to operate on this metadata representation in order to determine the metadata storage capabilities, for the representation and management, of a resource pool (library). The other part of the project is building the resource management part of the OntoAIMS application, using the ideas of the AIMS system and open standards for metadata, to manage and search the resource pool.

## *2.1 Research questions*

This section describes the research considerations and aspects thereof. This project will try to provide answers these research questions. The research questions form the base for the project goals.

Research question 1: How to reduce the complexity of searching for resources in the system for the author?

- 1.1 How to use the ontology (structure) to aid in searching resources.
- 1.2 How to visualize part of the (overall) OntoAIMS ontology, i.e. relations between OntoAIMS data objects (resource, concept, domain and course) and to show resources in context.
- 1.3 How to combine keyword search and ontological (relationship) browsing.
- 1.4 How to visualise common properties between resources in search results.

This question is further researched in paragraphs “3.5 Searching for resources” and “3.10 RMT Graphical User Interface”.

Research question 2: How to manage resources efficiently?

- 2.1 What management tasks to perform on resources?
- 2.2 How to use the ontology (domain/course/resource structure) to aid in managing resources.
- 2.3 How to position the resources in the overall OntoAIMS ontology; relating resources to other objects (resources/concepts/courses).
- 2.4 How to suggest helpful information to the user.

This question is further researched in paragraphs “3.2 Use Case” and “3.6 Architecture”.

Research question 3: How to model resources in OntoAIMS?

- 3.1 What to use of which open data standards.
- 3.2 How to use the semantics of the data.

This question is further researched in paragraph “3.3 Resources”.

## 2.2 Project goals

Derived from the research questions and the context for the project the following goals describe the intentions for the project:

From research questions 1, 2 & 3:

Goal 1 Create a Resource Management Tool (RMT), as a part of the OntoAIMS application, to manage resources in an educational system. The OntoAIMS educational toolkit is to replace / follow up to the AIMS application.

This goal is further researched in paragraph "3.6 Architecture".

From research question 1:

Goal 2 Create search strategy that uses the domain / course ontology to expand keyword search with results and relations derived from the ontology. The search algorithm will display the search results with (ontological) relationships and grouping by ontological relationships and by commonalities between the search result items.

This goal is further researched in paragraph "3.5 Searching for resources".

From research question 2:

Goal 3 Create a resource pool with management functions (create, edit, delete, view) on the resource objects in the resource pool. Guide the user in deleting a resource (inform of consequences) and creating/editing a resource (suggest information, offer information structures).

This goal is further researched in paragraph "3.4 Managing resources".

From research question 3:

Goal 4 Create a metadata specification for OntoAIMS resources based on selections from open standards for describing resources (in general) in such a way that:

- The OntoAIMS resource can be expressed in one or more open standard metadata specification(s) and
- The metadata for the OntoAIMS resource is limited to relevant descriptors.

This goal is further researched in paragraph "3.3 Resources".

## *2.3 Requirements*

The following definitions specify the requirements for the RMT application and the relations of the RMT application to the DMM and CST applications. The requirements are grouped by type.

### 2.3.1 Identification requirements

The identification requirements define the aspects of identifying the objects that are to be used (referenced to and described) in the RMT application.

1.1. An OntoAIMS resource is to be identified by its URI, with that URI other metadata elements of the identified resource can be requested from the resource pool. All OntoAIMS resources within the OntoAIMS application belong to the same resource pool.
1.2. An OntoAIMS (domain) concept is identified by its URI, with that URI other descriptors (metadata items) can be requested from the Domain Modelling Tool (DMT). Each (domain) concept belongs to a (knowledge) domain; each domain can have several concepts. The DMT manages several domains.
1.3. An OntoAIMS course is composed of selected (domain) concepts from a domain and selected resources from the resource pool. A string identifies an OntoAIMS Course.

These requirements are used for the design of the RMT in paragraph(s) “3.3.1 Resource modelling”.

### 2.3.2 Functional requirements

The functional requirements describe the functionality of the RMT application for the authors of the OntoAIMS authoring system.

2.1. The Resource Management Tool (RMT) is part of the OntoAIMS authoring environment. The OntoAIMS authoring environment is composed of the Domain Modelling Tool (DMT), the Resource Management Tool (RMT) and the Course Sequencing Tool (CST).
2.2. To use the Resource Management Tool the OntoAIMS authoring environment has to be running. There are however no requirements on the data currently used/opened in the Domain Modelling Tool or in the Course Sequencing Tool. The domain or course currently opened in the other editors has no effect on the management of resources in the RMT.
2.3. The set of OntoAIMS resources (the resource pool) is stored in RDF format.
2.4. A dictionary provides synonyms for English words that aid the RMT to extend search queries.

These requirements are used for the design of the RMT in paragraph(s) “3.9.1 OntoAIMS module communication” and “3.6.1 OntoAIMS structure” and chapter “4. Implementation”.

### 2.3.3 Data requirements

The data requirements define the minimal demands on the resource metadata.

3.1. A valid OntoAIMS resource has at least a URI, a name, a description, an author and a date.
3.2. A valid OntoAIMS resource requirement definition has a type, a name and a version identifier.
3.3. A valid OntoAIMS resource relation definition has a type, an URI of the target and a description.

These requirements are used for the design of the RMT in paragraph(s) "3.3.3 OntoAIMS Resource metadata".

### 2.3.4 Synchronisation / communication requirements

The synchronisation requirements specify the minimal agreements for interchanging information among the authoring applications of the OntoAIMS authoring system to maintain data integrity and informing the other applications of changes.

4.1. The Resource Management Tool notifies the Domain Modelling Tool and the Course Sequencing Tool when a resource is deleted; the DMT and CST then modify their data to stop referring to the deleted resource object.
4.2. The RMT notifies the DMT and the CST when the identification string, the URI, of a resources changes. The DMT and CST have to update their information to keep pointing to the right resource object.
4.3. Both the DMT and the CST inform the RMT of deletion of an object or a change in the identification string (URI) of an object to remove or update links to that object in the resource pool.
4.4. The RMT, DMT and CST can ask each other for the existence of a data item (resource, concept or course), the properties for a data item given its identifier and possible relations to other objects within the data of the authoring application.

These requirements are used for the design of the RMT in paragraph(s) "3.9.1 OntoAIMS module communication".

### 2.3.5 Authoring requirements

The authoring requirements specify the actions the author has to be able to perform on the system to work with the available (resource) data.

5.1. The user / author can create a new resource.
5.2. The author can delete an existing resource, if the resource is connected to other object(s) the author is to be warned.
5.3. The author can edit the properties / data elements of an existing resource, including the identification string (URI).
5.4. The author can view an existing resource. Resources can also be viewed from the DMT or the CST.
5.5. The author can connect a concept to a resource as a keyword relation when editing an existing resource or creating a new resource.
5.6. The author can connect an existing resource as a relation when editing an existing resource or creating a new resource.
5.7. The author can request to save the (changed) resource pool to disk. If the resource pool is changed and the author exits the application then the author is asked if he wants to have the data saved.
5.8. The author can search for resources using a keyword-based search by typing in (search) words in the search form and by browsing data and relations between data in the graphical elements of the user interface.

These requirements are used for the design of the RMT in paragraph(s) “3.2 Use Case”, “3.4 Managing resources”, “3.5 Searching for resources” and “3.9.2 RMT sequence diagrams”.

### 2.3.6 Graphical requirements

The graphical requirements define the aspects of the graphical user interface environment for the author to work in.

6.1. The main menu structure supports separate sections per authoring application and some general sections. The general sections are enabled when the total OntoAIMS application is running. The specific menu sections are enabled when the appropriate authoring component has focus and are disabled otherwise.
6.2. The main user interface of the Resource Management Tool (RMT) component of the OntoAIMS application consists of five parts:
    a. A search form where the author can formulate and submit a search request (query).
    b. A hierarchical navigation pane (the 'cluster tree'), displaying the hierarchical structure of the clustering of search results.
    c. A list (the 'resource list'), which contains the set of resources from the search results that belong to the currently selected cluster.
    d. A structure navigation pane (the 'relation graph' or the 'graph'), visualising the relations (to a relation depth of 1) of the currently selected[3] object[4].
    e. A description pane, providing a description of the selected object[4].
6.3. Selecting an object[4] in the list, tree or graph updates the other view elements (including the description pane) and forces the same selection[5] in the other view elements.
6.4. The DMT provides a user interface dialog to select a domain concept (URI) to use in the metadata of a resource.
6.5. The RMT provides a user interface dialog for the DMT and CST to search for resources to use in the metadata of a concept or course.
6.6. The RMT provides a user interface dialog for the DMT and CST to view the metadata of a resource.
6.7. Information dialogs are shown to ask the author for confirmation on deleting objects, saving information, warnings, errors, etc.

These requirements are used for the design of the RMT in paragraph(s) "3.10 RMT Graphical User Interface".

[3] The currently selected object is the object placed in the centre of the graph.
[4] The object in the graph can be a resource, concept or cluster.
[5] This selection will only be set when possible. Since the 'tree' only contains clusters, the resource list only contains resources and the graph can contain all three of the object types. If a selection in 'tree' or 'list' is not possible the selection will revert to none for that window part.

# 3. Development of the Resource Management Tool

In this section the different stages of designing the structure, logic and look and feel of the OntoAIMS Resource Management Tool are presented. From the project goals and requirements the main tasks (use case), subtasks and data structures (resources) are deduced. From the authoring tasks an architectural structure is formed. Upon that structure (architecture, classes, sequences), with the requirements in mind, the author-system interaction and visualisation is defined.

## *3.1 Differences with AIMS*

The OntoAIMS Resource Management Tool resembles (part of) the AIMS system. The main differences with the resource management in the AIMS system will be:

- All the resources in the OntoAIMS RMT are available to all concepts in all domains and to all courses. In AIMS a set of resources belonged to a domain and that domain was used to make up a course. This way the resources are of a diverse nature and are reusable across domains and courses. The presentation and manipulation of resources needs to be domain and course independent.
- In OntoAIMS the separation between the Course-tool, the Domain-tool and the Resource-tool is very strict. Each tool is responsible for maintaining its own data and communicating changes and requests with the other tools.
- In OntoAIMS we use open web standards; resource annotations are defined using an open standard and are represented using RDF. Domain models are represented using OWL. Course sequencings will be SCORM-compliant. The advantage of using open web standards is that it opens the opportunity to exchange information with other systems that use such standards. This can significantly improve the usability of the system, by speeding up the authoring process. Instead of defining resource annotations and building domain models themselves, instructors can use already available resource annotations or general domain ontology's (and alter these to their specific needs).
- The OntoAIMS modules will provide more views on the data to further explore relations and information.

## *3.2 Use Case*

[UML]: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures. Associations between actors and use cases are indicated in use case diagrams by solid lines. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

The Use Case model for the RMT is shown in Figure 8. It describes the authoring tasks the user can perform on the system, the subtasks composing the task and the effect (extend) the task has.

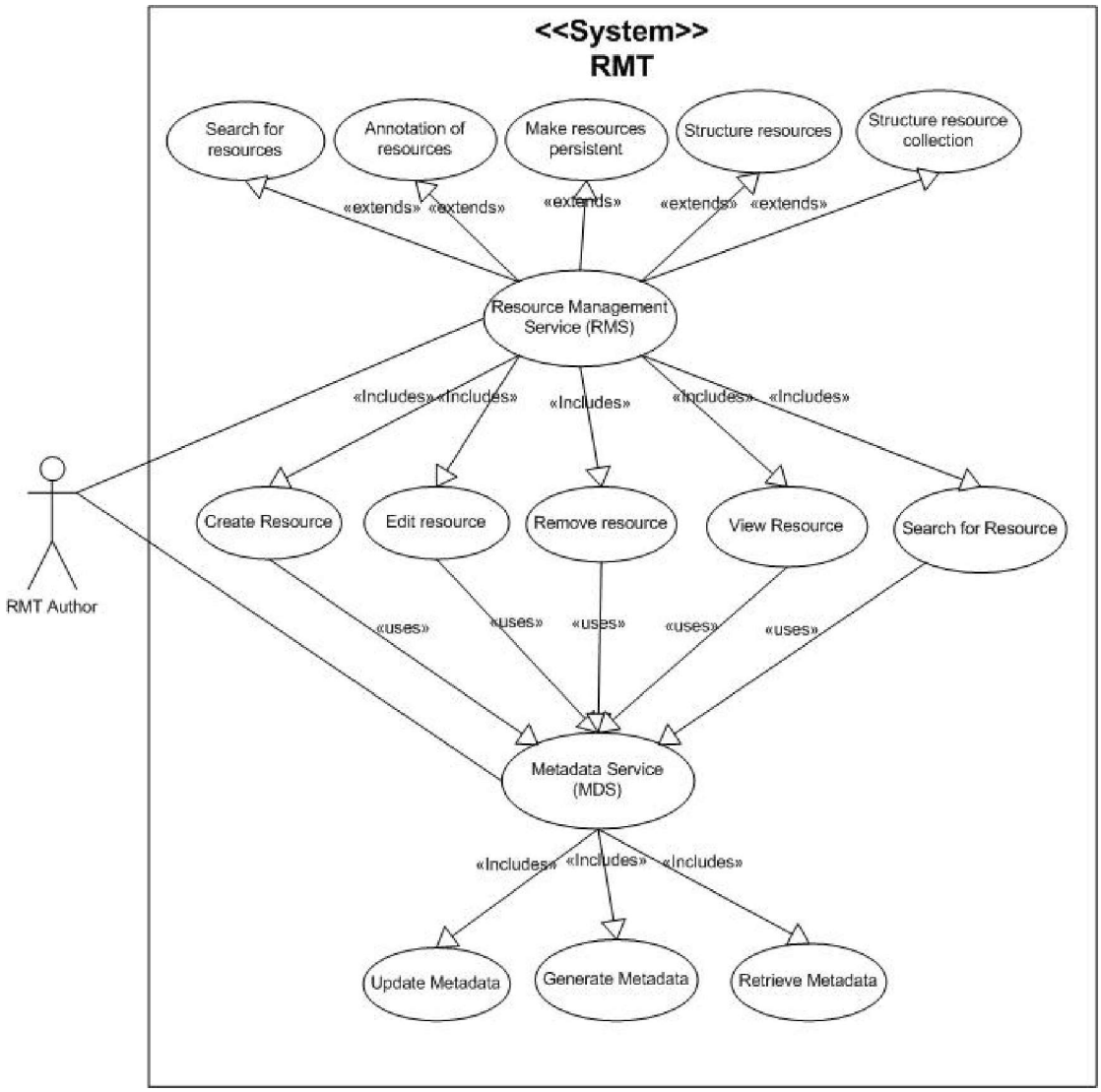


**Figure 8, RMT main Use Case**

The author uses the system to accomplish resource management using a group of tasks, collectively called Resource Management Service, which is supported by a Metadata Service. The tasks the author can perform on the system are:

- Create a resource (add a resource to the resource pool)
- Edit a resource in the resource pool
- Remove a resource from the resource pool
- View the metadata of a resource in the resource pool
- Search for resources in the resource pool

These tasks have the following effects on the (data of the) system:

- Create structure in the resource collection (pool)
- Structure resource metadata
- Make resources persistent
- Annotate resources
- Search for resources

The inner workings of the tasks the author can perform rest upon the capabilities of the Metadata Service, which are:

- Update the metadata for a (specific) resource in the resource pool
- Generate (create new) metadata for a resource in the resource pool
- Retrieve the metadata for a (specific) resource in the resource pool

The Search for resources task that the author can perform can be elaborated on, the search for resources is composed of:

- Search by traversing relations between objects through the system; browsing
- Search by entering search words / keyword(s) and have the system match those keyword(s) to resource descriptions

The search algorithm will use a dictionary to extend the keyword search and, as said before, the Metadata Service.
These details of the subsystem are shown in Figure 9.

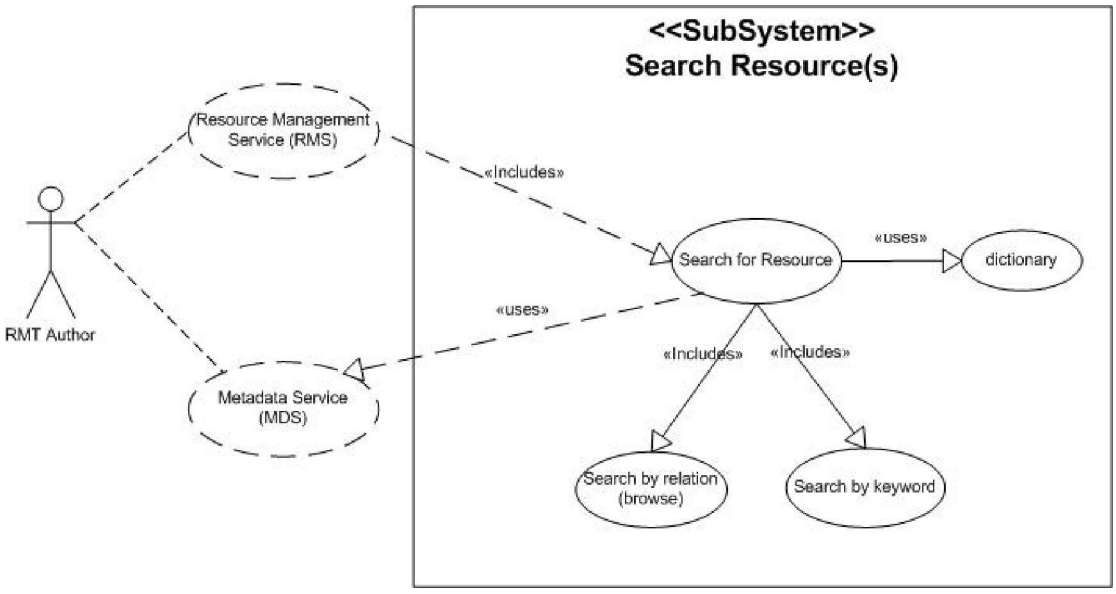


**Figure 9, RMT sub Use Case – Search**

## *3.3 Resources*

The main data component / data structure in the RMT is the resource. A resource in the RMT is a collection of metadata describing and identifying a resource object. Resources can anything that can be identified in the real or virtual world. I.e.: it can be a HTML webpage, a learning system, a person, a book, a magazine, etc.

### 3.3.1 Resource modelling

OntoAIMS adopts open (web) standards for resource annotations to store its information. A suitable format has to be selected.

To uniquely identify a resource, concept or course in the system, each object needs its own identifier number or name. For this purpose all resources, domain concepts and courses are referred to by their URI. See also [GBIV] and Appendix 4: URI.

In an early attempt to specify metadata items to describe a resource the identification of such a resource was realised by an internal number. Since import and export functions may need costly renumbering algorithms to prevent duplicate identification numbers and each resource has a unique URI anyway the numbering scheme was discarded.

The URI that identifies a resource is the location of that resource on the Internet / in a library / etc. This URI may change over time because the location of the resource moves / restructures / the editor finds a copy of the resource on another location or the location was mistyped when the resource was entered into the system. When the identification of a resource changes the other authoring systems need to be notified of this change.

For further specifications we adopt the following primitives to describe the objects in the OntoAIMS resource management tool:

Resource is the modelling primitive that represents the collection of metadata describing one (learning) object. A resource can describe anything that can be identified with an URI. Resource information is specified using attributes.

Domain Concept is the representation in the Resource Modelling Tool of a concept in a domain in the OntoAIMS Domain Modelling Tool (DMT). It contains the information to identify the concept (and its domain) in the set of domains managed by the domain editor. Concepts are not further annotated because they are already annotated and managed in the OntoAIMS Domain Modelling Tool (DMT).

Cluster is a named container for grouping the resources in search results to reflect commonalities over that group of resources.

Requirement is a property of a resource to specify that the learning object which the resource describes, has a requirement from the author or system to view or use the learning object. A resource can have 0 or more requirements.
Example: resource is of type ‘PDF file’ and therefore needs ‘acrobat reader’

Relation is a property of a resource to specify that that the learning object which this resource describes, has a relation to another learning object described using a OntoAIMS resource. A resource can have 0 or more relations.
Example: a resource is a chapter from a book and has a relation to the resource of the book that contains that chapter.

For two of the OntoAIMS editors, the Domain Modelling Tool (DMT) and the course sequencing tool (CST) the specific use of metadata specifications was determined before work started on the Resource Modelling Tool (RMT).

For the Course Sequencing Tool the Sharable Content Object Reference Model [SCORM] specification was chosen. And for the Domain Modelling Tool the specifications of Web Ontology Language [OWL] were selected.

Because of the shared characteristics with SCORM and OWL the specifications of Learning Objects Metadata [LOM] were chosen for the Resource Modelling Tool to use to describe resources.

LOM is a conceptual data schema that defines the structure of a metadata instance for a learning object. For this Standard, a learning object is defined as any entity -digital or non-digital- that may be used for learning, education or training.

A metadata instance for a learning object describes relevant characteristics of the learning object to which it applies. LOM supports descriptions on several topics such as general, life cycle, meta-metadata, educational, technical, rights, relation, annotation, and classification categories. The standard does not define how a learning system uses or represents metadata for learning objects but helps structuring data.

*"The purpose of this multi-part Standard is to facilitate search, evaluation, acquisition, and use of learning objects, for instance by learners or instructors or automated software processes. This multi-part Standard also facilitates the sharing and exchange of learning objects. By specifying a common conceptual data schema, this Standard ensures that bindings of Learning Object Metadata have a high degree of semantic interoperability. This Standard specifies a base schema, which may be extended as practice develops."* [LOM]

Since the specifications of Dublin Core [DC] and LOM have a large overlap the OntoAIMS resource descriptions shall be in LOM but wherever a metadata description field (property) is available in LOM and in DC the DC 'label' will be used. This gives the metadata (the resource pool) the most reusability and backward compatibility.

In addition to the metadata descriptions available in DC and in LOM more information fields are added to support use of OntoAIMS specific data (i.e. referencing a domain concept).

### 3.3.2 More metadata description standards

For the DMT and the CST the metadata specifications ware determined before work started on the RMT. For the CST the SCORM specification was chosen and for the DMT the specification of OWL was selected.

Next to LOM and DC much more open standards for describing resources exist. Here are some of those other open metadata standards and why they did not fit this project (besides the reason mentioned earlier in this paragraph):

SCORM. The Sharable Content Object Reference Model [SCORM] by Advanced Distributed Learning (ADL) is a very large specification for designing a SCORM runtime environment. The specification can be used as a reference set to describe metadata but SCORM focuses on making 'Content Aggregation Models' to implement whole systems in its own environment. For the RMT only the aspect of describing metadata of SCORM could be useful but SCORM offers less resource description elements then LOM.

IMS Meta-Data [IMS] by IMS Global Learning Consortium
The IMS specification offers an extended set of metadata descriptors above the LOM and DC specification. It offers specifications for describing learner process tracking, reporting and exchanging student information. All these added description specifications have no added value for the RMT since the RMT does not interact with students; they are tracked in the viewer application and not in the authoring application.

Gateway to Educational Materials [GEM] is much like the LOM specification for metadata definition. The GEM specification has less metadata description items than LOM and therefore offers no added value to the RMT above LOM.

The Cisco Reusable Information Object [RIO] initiative defines a procedure for the development and delivery of training modules called Reusable Learning Objects (RLOs). Cisco RLO's are tagged with additional Metadata indicating each object's placement on a cognitive continuum that extends from static knowledge (memory) to increasing degrees of applied knowledge, such as concept, fact, procedure, principle, and process. This system is not suited for the OntoAIMS RMT because the strict structure of $5+2$[6] learning items that build up a RLO. This structure has no use for the RMT because there are no tests in resources.

The metadata description sets mentioned above are not all standards available. There are even more standards, specifications and proposals available like:
Canadian Core Learning Resource Metadata Protocol (CanCore Protocol) [CanCore], Metadata Object Description Schema [MODS], Machine-Readable Cataloguing [MARC] or Content Standards for Digital Geospatial Metadata [CSDGM].

[6] Each RLO has to contain 5 learning items and 2 tests. This combination forms a learning task.

### 3.3.3 OntoAIMS Resource metadata

The resource metadata for the OntoAIMS Resource Management Tool is based on the specifications for resources in AIMS (there they are called documents in libraries) and (parts of) one or more open standards for metadata definition. The next paragraphs sum up the selected metadata definition properties.

3.3.3.1 From AIMS

In AIMS the library for a course and a domain has the following properties[7]:
*Code, Name, Author, Description, Create date, Last modify date*

Each document in an AIMS library has the following properties:
*Name, Author, Year, Location, Presentation format, Instructional format, Description, Keywords (+ weights), Create date, Last modify date.*

The AIMS Presentation format and Instructional format have a limited set of values (vocabularies):

| # | Vocabulary | Members |
|---|---|---|
| A | Presentation format | Audio, Diagram, Exe file, Text, Video, Zip File |
| B | Instructional format | Assignment, Book, Case, Chapter, Description, Example, Exercise, Journal, Lecture, Manual, Paper, Procedure, Questionnaire, Reader, Test. |

OntoAIMS has one resource pool for all domains and all courses, in contrast to the structure of AIMS where one course, one domain and one library form a set.

3.3.3.2 Looking into open standards

DC has these descriptors:
*Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights.*

LOM has many descriptors like:
*Identifier, Catalog, Entry, Title, Language, Description, keyword, Coverage, Structure, Aggregation level, Version, Status, Contribute, Etc.*

LOM descriptors are grouped. The groups are named:
*General, Life cycle, Meta-metadata, Technical, Educational, Rights, Relation, Annotation and Classification.*

In OntoAIMS the main interest in resource annotation is educational data. For this reason the focus on the LOM descriptors lies most with the groups General, Technical and Education. For the rest of the selection from this standard the individual tags (descriptors) are evaluated on usability for the RMT.

[7] Over specification of standards and texts on educational systems several terms are used for metadata identifiers for describing data: ‘descriptors’, ‘tags’, ‘fields’, ‘description properties’, ‘properties’, etc.

3.3.3.3 Selecting the OntoAIMS resource set

By combing the tags that are used in the AIMS system with those in DC, a base set of descriptors for the OntoAIMS resource is created. This set is then expanded with a selection from the tags from LOM that have a meaningful use in describing OntoAIMS resources. When there are more tags that describe the same data, then the tag will be written in the first preferred standard available. The preferred order is: DC > LOM > Rest.

The gathered set of metadata will be used to manage resources in the OntoAIMS resource pool and by its structure can be easily distributed as a DC or LOM data structure to other tools or systems.

The metadata descriptors for the OntoAIMS resources are specified as follows:

| NS: Tag: | Dc: Date | Description: | Date on which the resource was created by the author |
|---|---|---|---|
| Name: | Creation date | Example: | "May 2000" or "27-03-1922" |

**Table 1, metadata element example - Date**

The tag "Date" from DC is equivalent with "Year" from an AIMS Document and "Date" from LOM.

The full list of metadata descriptors for OntoAIMS resources is specified in Appendix 1: metadata tags.

The relational structure between the metadata descriptors is visualised in a conceptual model. The conceptual model can be found in Appendix 3: Conceptual model.

## 3.3.4 OntoAIMS data relations

The OntoAIMS data objects have references to each other to denote relationships between the data. The 'connectivity' between the data objects is visualized in the following figures:

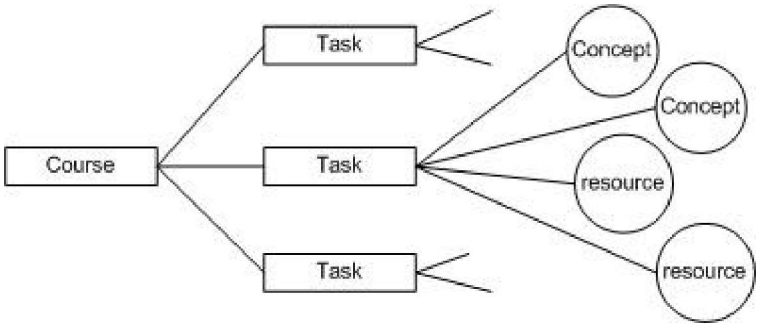


**Figure 10, data dependencies for courses**

Figure 10 shows the relations for courses. Each course has a set of course tasks. Each course task references concepts and resources to describe the learning subjects that the course task covers.

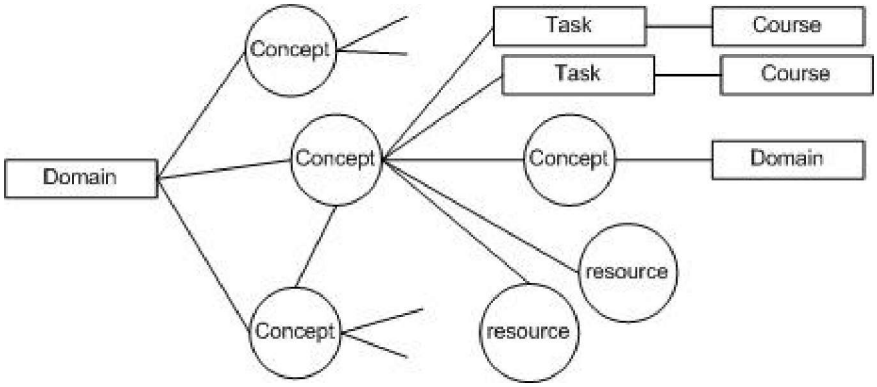


**Figure 11, data dependencies for domains**

Figure 11 shows the relations for domains. Each domain has a set of concepts. Concepts have references to other concepts; describing equivalence, parent-child relations, sharing of information, etc. Concepts are referenced in course tasks. Concepts have references to resources (that describe the concept).

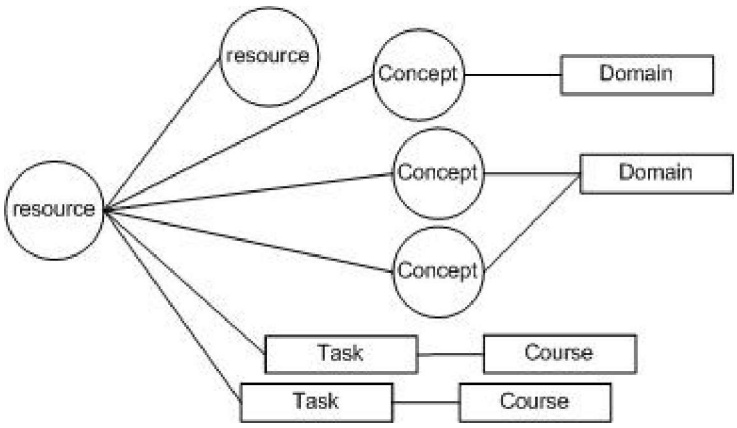


**Figure 12, data dependencies for resources**

Figure 12 shows the relations for resources. Resources reference concepts to describe the resource. Resources are referenced by concepts to (partly) describe the concept. Resources are referenced in course tasks. Resources have references to other resources; describing equivalence, parent-child relations, etc.

## *3.4 Managing resources*

In paragraph 3.2 the basic operations on resources are defined. Next to these operations on a single resource object the system offers the operation of searching through the whole resource pool, this functionality is described in the next paragraph.

It is important to realise that the modification of a resource can have an impact on other parts of the system. For instance deleting a resource in the resource pool makes links in a domain model to that resource invalid. A concept may be linked to one or more resources and the manager of that concept has to be notified of the removal of the resource from the resource pool to update (remove) the link from its data and maintain data integrity.

The same goes for changing resource metadata that is connected to: a resource can have links to courses or domain concepts and changing the identifier of a resource has a similar impact to removing the resource from the resource pool. For this reason every important change to the resource pool has to be reported to the DMT and the CST and both the DMT and the CST have to report larger changes in their data to the RMT.

When creating / editing a resource the authoring requirements (paragraph 2.3.5) prescribe that:

- All metadata elements of resource may be changed / edited,
- The author has to be warned when deleting a resource from the resource pool and
- The author has to be informed if the resource he wishes to delete has links to other objects (course task / domain concept) in the system.

## *3.5 Searching for resources*

Next to managing resources the searching for resources is a main task of the resource management system. The user (resource author) needs to be able find the resources he manages and not just by already knowing what to look for.

To aid the author in enhancing the chance of finding what he is searching for the system adapts the author's query by adding information that may give related results and remove 'noise'. Noise in a query is the use of common words, stop words and words that join sentences. Noise words in a query are likely to give results that match on words but not on the meaning that the author is searching for.
Examples of noise are: 'a', 'and', 'be', 'by', 'or', 'then', 'else', 'this', etc.

### 3.5.1 Sorting versus searching

The RMT can contain resources on all kinds of subjects. When the author wants to see resource on a certain subject these resources can be displayed to the author in 2 ways:

1) By searching for resources. The author will then get an overview of the subset of resources from the resource pool that match the requested subject or search word(s).
2) By sorting all the resources in the resource pool. The author is presented with the list of all resource in the resource pool ordered by relevance to the requested subject or search word(s).

Since we want to see more (ontological) relations, between the data that the author requests, searching is chosen over sorting. This visualisation of ontological relations is further specified in *paragraph 3.5.5 Clustering search results.*

### 3.5.2 Filtering the user query

The first step towards useful search results is analyzing the query posted by the author to the system and modifying this query to get as many relevant documents as possible. We try to eliminate 'noise' from the query, add equivalent words to search words and incorporate previously made relations within the system.

The strategy for filtering 'noise' from the query is based on the working of the search engine Google [Google] and a presentation on *Query modification and relevance feedback* [Query].

Good search queries can be difficult to create and authors usually know some words of what they are looking for but not the precise title or name. To aid the author in his searching the system can help the author in filtering and enriching the search query.

The process of transforming the author's query is composed of several steps. Each of the steps shows the transformation of an example query in a logical formula notation[8]. The 'first step' is receiving the query from the author: step 0.

Step 0:

| | |
|---|---|
| | {Let Q be a user query, let A, B, C be (search) words (separated by white spaces)}<br>Q = A B C |

The user entered a search query into the system by typing words into an input field and pressing enter or clicking on a submit button. Such a query might be "*cold fusion reactor*" or "*java coffee beans*" (the user enters the query without quotes).

To analyze / filter the words of the query separately the query is split into search words: Step 1 shows the breaking of the query on the white spaces:

Step 1:

| | |
|---|---|
| | Q = A B C |
| à | {split query into (search) words} |
| | Q = {A, B, C} |

With the splitting of the user query in separate search words each part (each word) can be filtered / expanded.

Next 'noise' is removed from the query. 'Noise' is any string of characters that is likely to give bad search results (false positives) or that is likely to have no search results and therefore only costs time to process.

[8] These steps do not form a proof but show the transitions between states of the query although a mathematical notation is used.

To filter out as much 'noise' as possible from the (split) user query several filtering actions are taken:

- Search operators such as "+", "and", "or", etc.[9] are removed.
  For each search all words of the search are evaluated separately and an intersection of those separate results is also created. This way both the results for 'or' and the 'and' search mechanism on the query are presented to the user. This way no grammar is imposed on the user for searching.
- Common words, words that occur in most texts without adding much meaning, are removed. Use of common words like "for", "with", "then", "else", "if", etc. are likely to give search results that have no value to the rest of the query.
- Short words, words that are less than 3 characters in length, are removed. Short words are likely to be part of longer words in texts and those longer words are very likely to have a complete different meaning.

Step 2 shows the actions in filtering:

Step 2:

| | |
|---|---|
| | Q = {A, B, C} |
| à | {Filter out common words / reserved words / words with less than 3 characters} |
| | Q1 = {A, B, C} – {filter words} |
| = | {Let A, C not be element of {filter words}, let B be element of {filter words}} |
| | Q1 = {A C} |
| à | {Rewrite query from collection form to logic form} |
| | Q = A \/ C |

After filtering only the words that are likely to give results are left in the set of search words.

### 3.5.3 Query expansion

After filtering more relevant data is added to the 'new search query'. The relationships between the OntoAIMS data items are used to add 'already known' or 'ontological information' to the search query. For this reason each of the filtered search words is matched to concept names in the knowledge domains in the DMT. With this information the query can be expanded with:

- Concepts, which are defined by an 'equivalence relation' in a knowledge domain. An author has already defined those concepts to be equivalent and therefore help in expanding the search query with other words to get more similar results.
- Resources that already have a relation specified with concepts with the same name as the search word (or equivalent concepts). These resources can be added to the search result for the search word and may otherwise not be found by just matching search words to descriptions of resources.

Step 3 shows the steps actions in extending the query with ontological information:

[9] There is one exception: placing more then one word between double quotes (""), with that the string of characters between the double quotes will be handled by the system as 1 search word.

Step 3:

| | |
|---|---|
| | Q = A \/ C |
| à | {Expand with domain concepts matching the name of the search word (retrieved from the OntoAIMS Domain Modelling Tool).<br>Ac is the set of concepts for A with length N: Ac = Ac1 \/ Ac2 \/ … \/ AcN<br>Cc is the set of concepts for C with length X: Cc = Cc1 \/ Cc2 \/ … \/ CcX } |
| | Q2 = (A \/ {Ac}) \/ (C \/ {Cc}) |
| = | {Let A' be "A \/ Ac1 \/ Ac2 \/ ... \/ AcN", same for C} |
| | Q2 = A' \/ C' |

Next to adding ontological information to the query even more equivalence relations can be added tot the query by adding equivalent words for the search words. To get equivalent words a dictionary is used: the WordNet lexical reference system [WordNet]. This adding of synonyms works towards more search results with a similar meaning of the query that the author posted to the system.
Step 4 shows the actions in adding synonyms to the search words:

Step 4:

| | |
|---|---|
| | Q2 = A' \/ C' \/ (A' /\ C') |
| à | {Expand with synonyms retrieved from the data structure of WordNet.<br>Let As be the set of synonyms for A with length M: As = As1 \/ As2 \/ … \/ AsM<br>Let Cs be the set of synonyms for C with length Y: Cs = Cs1 \/ Cs2 \/ … \/ CsY} |
| | Q3 = (A' \/ {As}) \/ (C' \/ {Cs}) |
| = | {Let A'' be "A' \/ As1 \/ As2 \/ .. \/ AsM", same for C} |
| | Q3 = A'' \/ C'' |

The addition of synonyms is only done with the synonyms of the search words and not those of equivalent concept names found in step 3. It was considered likely that:

- The synonyms of equivalent words of search words are the same as the synonyms of search words or
- If the equivalent term is very different from the search word that synonyms of that equivalent word differ too much from the search word.

The 'new search query' is now likely to have the following effects:

- Less bad search results because of noise filtering
- More relevant search results because of adding already known relations/data
- More relevant results because of adding more meaning to query

### 3.5.4 Retrieving resources for the search

With the extended search query constructed in the previous paragraph the resources (documents) can be gathered. The filtered and extended query consists of groups of related words; each set of words expresses a filtered and extended part of the original query by the author. The combination of {filtered search word, matching concepts for that search word and synonyms for that search word} are considered 1 'search item'. Documents that have at least one of the words in a 'search item' in its description are considered a match to that 'search item'.

If a user (author) enters a query comprised of a single search word, and this word does not get filtered out as 'noise', the query consists of 1 'search item' after extending. When the extended query has more then one 'search item' we are interested in the document results for each 'search item' and the document results that the 'search items' have in common.

Several search systems (like SQL or Internet Search Engines) use the words 'AND', 'OR' and 'NOT' to let the user refine or widen the search range with other search words. The effect of the 'AND' and 'OR' keyword is illustrated in the Venn diagrams of Figure 13. In the RMT the 'OR', 'AND' and 'NOT' keywords are filtered out of the search query to eliminate them as 'noise' and because both the 'AND' relation over all search items and all separate search items will be displayed to the user.

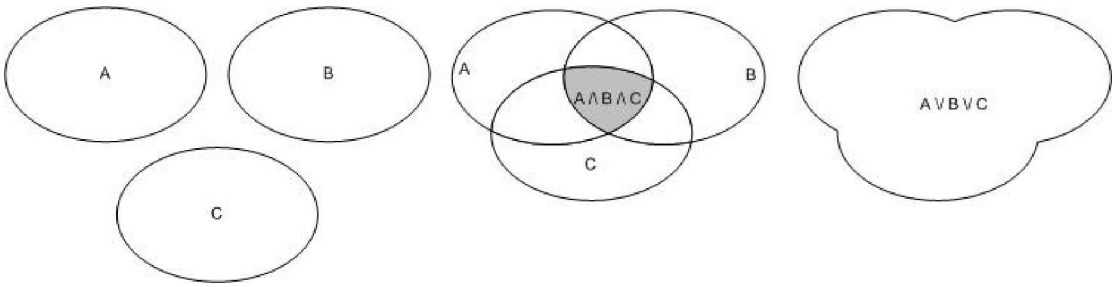


**Figure 13, Venn diagrams**

The Venn diagrams in Figure 13 show (from left to right):

- The 3 result sets of 3 search items (A, B and C),
- The 'AND' relation (grey area) over the 3 search items and
- The 'OR' relation over the 3 search items

Example:
An author query could be: "the car maintenance manual".
After filtering and extending this query could be something like: ({car, auto, Ford, Toyota}, {maintenance, service}, {manual, book, instructions})
This can be mapped to Figure 13 as:
A = resource results for ("car" or "auto" or "Ford" or "Toyota")
B = resource results for ("maintenance" or "service")
C = resource results for ("manual" or "book" or "instructions")

In the OntoAIMS RMT both the 'AND' relation over the resource results of all search items and the resource results for each search item are displayed to the author.
Figure 14 shows the partial resource result sets for a query.

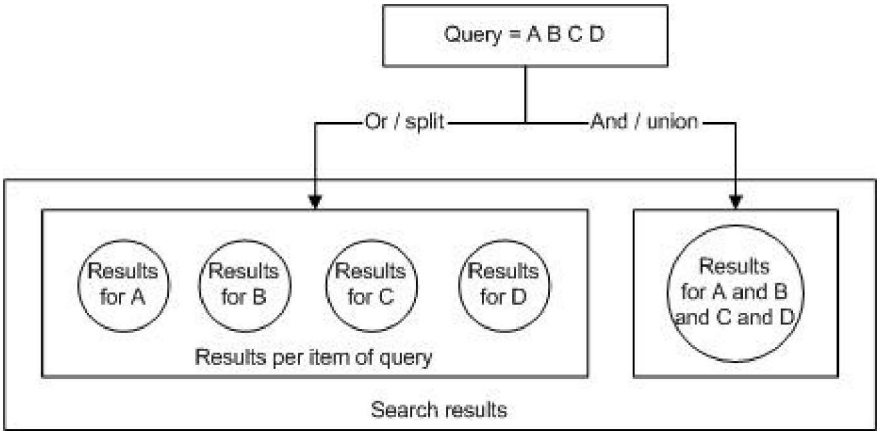


**Figure 14, views of query parts**

Step 5 shows the actions in building the sets of document results.

Step 5:

| | |
|---|---|
| | Q3 = A'' \/ B'' \/ C'' |
| à | {Get all documents for query Q3} |
| | D1 = All documents where any word in A'' exists in the description of the document.<br>D2 = All documents where any word in B'' exists in the description of the document.<br>D3 = All documents where any word in C'' exists in the description of the document.<br>D0 = All documents that are in D1 and in D2 and in D3 (D1 intersect D2 intersect D3) |
| = | {Write as collection} |
| | {D0, D1, D2, D3} |

All document result sets are lists of identifiers of metadata descriptions of resources. Each identifier is an URI that represents an RMT resource (set of metadata). Documents for a 'search item' are found in the resource pool by matching all search words of a 'search item' to the metadata fields of a resource. If any of the words of a 'search item' is found in the metadata then that document is part of the result set for that 'search item'.

### 3.5.5 Clustering search results

When the resource author searches for keywords the direct result is a (possibly large) list of resource names without further structure.
Grouping the resulting set of resources from a search query into clusters creates:

- Overview of commonalities between resources
- Viewing of data per subject / per part of the query
- Simpler view of the data

Since the result sets for each 'search item' are kept apart during the gathering of documents these groups can be easily converted to the base clusters. Each 'search item' and the cross-section of the 'search items' get a base cluster. Step 6 shows the forming of the base clusters.

Step 6:

| | |
|---|---|
| | Q3 = A'' \/ B'' \/ C''<br>D1 = All documents where any word in A'' exists in the description of the document.<br>D2 = All documents where any word in B'' exists in the description of the document.<br>D3 = All documents where any word in C'' exists in the description of the document.<br>D0 = All documents that are in D1 and in D2 and in D3 (D1 intersect D2 intersect D3) |
| à | {Make base clusters} |
| | Cluster C0 = D0<br>Cluster C1 = D1<br>Cluster C2 = D2<br>Cluster C3 = D3 |

The search result set of resources is clustered in several ways. By grouping all resources from the:

- Search results that are used in the same domain.
- Search results that are connected to the same concept, either by keyword relation in a resource description or by a resource relation in a domain.
- Search results that share a commonality. E.g. resources that have the same type, resources that have the same format, resources that have the same author, etc.
- Search results that were found using the same 'search word'.

Step 7 shows that each base cluster is subdivided into sub-clusters. Each base cluster maintains its set of document results and the documents matching a sub cluster are copied to that sub cluster.

Step 7:

| | |
|---|---|
| | {Base clusters} |
| à | {Make sub clusters} |
| | C0<br>\|--C1<br>\| \|-- Sub 1 of C1<br>\| \|-- …<br>\| \-- Sub M of C1<br>\|--C2<br>\| \|-- Sub 1 of C2<br>\| \|-- …<br>\| \-- Sub K of C2<br>\|-- Sub 1 of C0<br>\|-- …<br>\-- Sub N of C0 |

A text example of the steps in searching and clustering is shown in Appendix 8. Chapter '3.10 RMT Graphical User Interface' describes the user interface that supports the author (user) in browsing the clustered search results.

"Appendix 8: RMT Search example" shows an example of searching, gathering and clustering of resources.

### 3.5.6 Search overview

The previous 4 paragraphs describe the steps in searching for resources and grouping search results. This process is schematically shown in Figure 15.

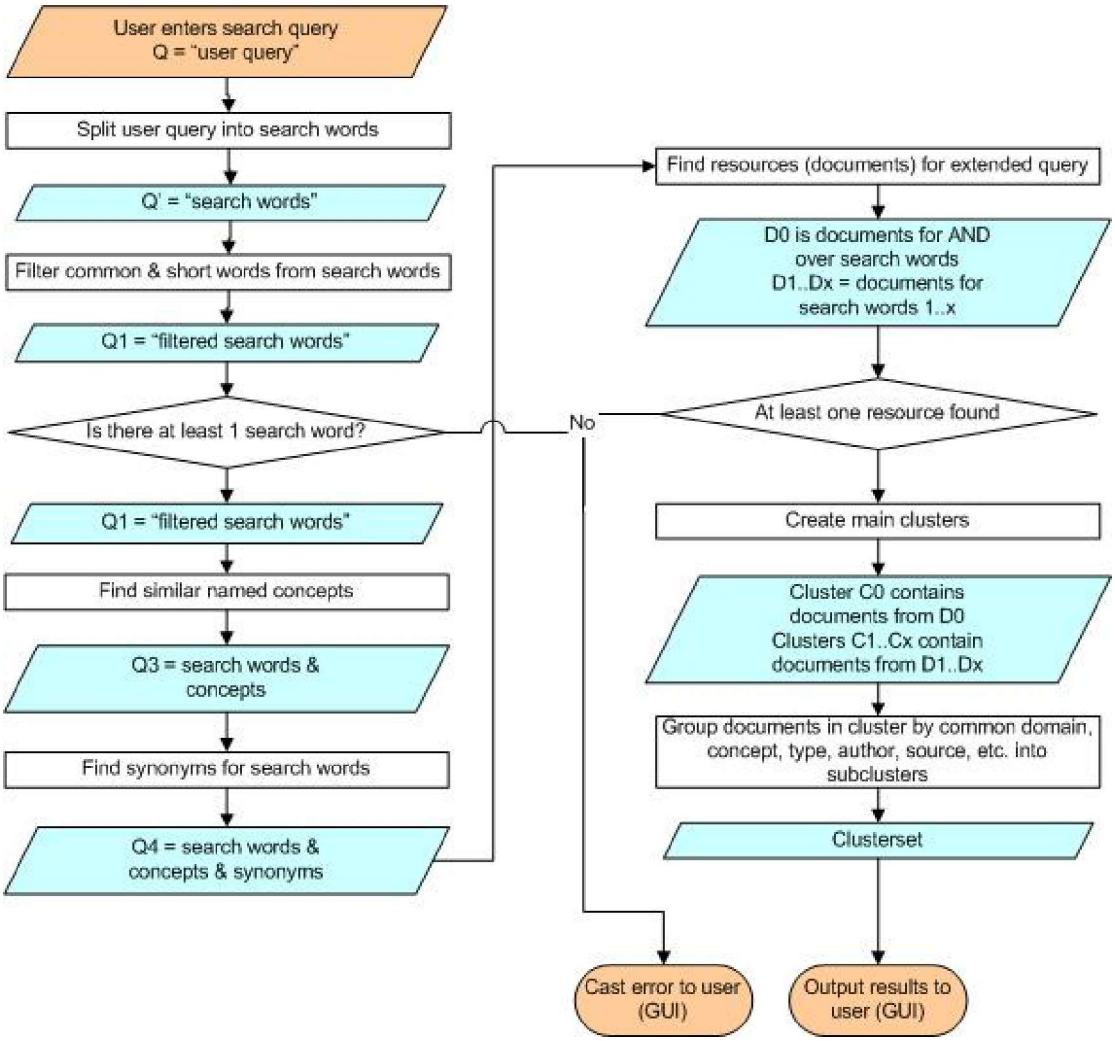


**Figure 15, schematic of searching for resources**

Figure 15 shows the steps in searching for resources. These steps describe the processes of:

- Query filtering,
- Query extending,
- Gathering of metadata objects and
- Clustering search results.

## 3.6 Architecture

Within OntoAIMS, as it was in AIMS, three different types of information are the focus of the system:

- Resources, the objects that a student can learn from
- Domains, collections of concepts that the author uses to describe knowledge on a subject
- Courses, tasks to be followed by the student in order to learn

### 3.6.1 OntoAIMS structure

The OntoAIMS authoring system is composed of three authoring modules. The DMT manages domains, the CST manages courses and the RMT manages resources. These authoring modules need to share data from and information about the dataset they manage. The modules and datasets are visualized in Figure 16.

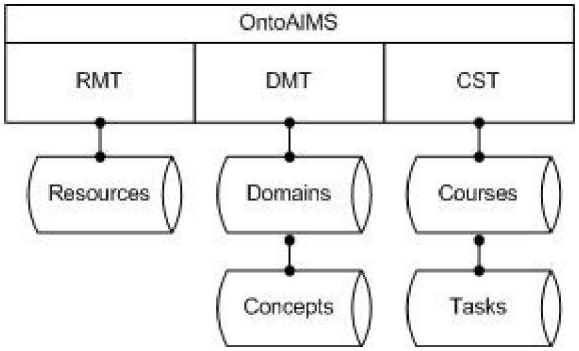


**Figure 16, OntoAIMS modules and their datasets**

The RMT is to be built within this separation of OntoAIMS in 3 data management parts, the communication between those parts and taking into account the project goals and requirements of paragraph 2.2 / 2.3.

The OntoAIMS author interacts with each of the three authoring modules as shown in Figure 17.

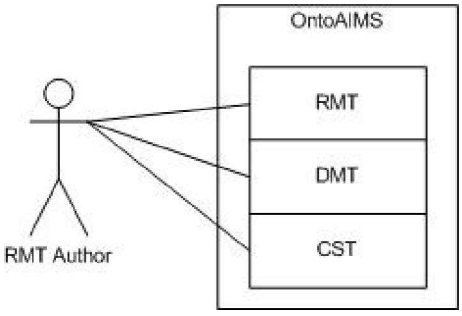


**Figure 17, the OntoAIMS authoring parts that the author interacts with**

For the purpose of better searching of data in any authoring module an additional module is added to the system to provide synonym information. The dictionary module provides this synonym information to the authoring modules. This module does not interact with the author but is only used by the other authoring modules. The components of the authoring system are shown in Figure 18.

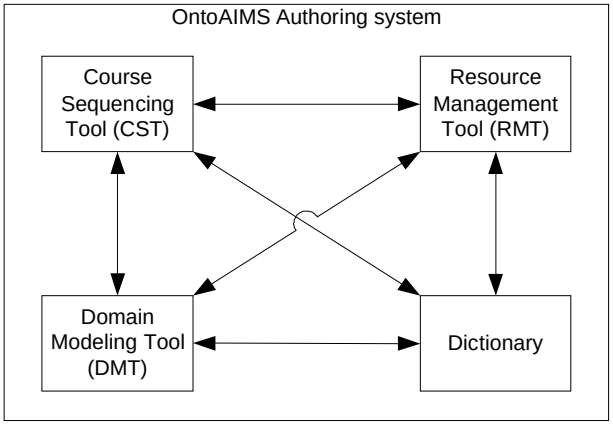


**Figure 18, Authoring module interaction**

Each authoring module is responsible for its own data set and the use of data from another authoring module always done by referencing a data object's URI. Using this unique identifier an authoring module can always request more information about the data object (resource/concept/etc) from the managing authoring module.
The communication between the modules is described in paragraph 3.9.1 OntoAIMS module communication.

To specify what requests for data and messages (update/delete/etc) to pass to and from an authoring module each authoring module has an interface specification. The components with interfaces are shown in Figure 19.

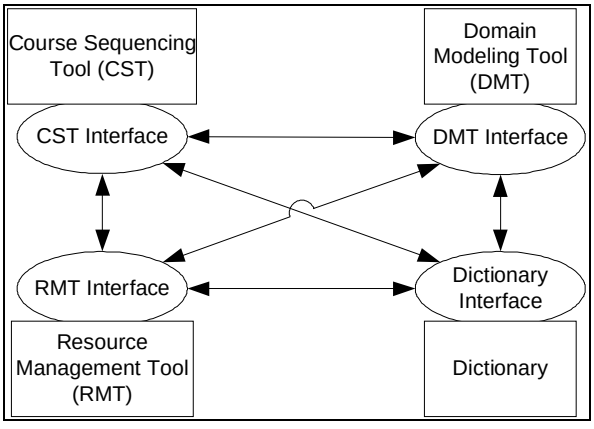


**Figure 19, Authoring module interaction interfaces**

Using these interfaces the authoring modules report changes in their data set, ask for information on a data object or search for data objects. I.e. the DMT requests a list of resources from the RMT. Or the RMT requests the equivalent concepts for a concept The operations supported by the interfaces are specified in paragraph 3.9.1 OntoAIMS module communication and Appendix 7: Module communication.

### 3.6.2 Shifting focus to the RMT architecture

Figure 20 shows the relevant parts of the system to describe the RMT and its interactions. The figure is an abstraction from Figure 19 with the user interaction added to it.

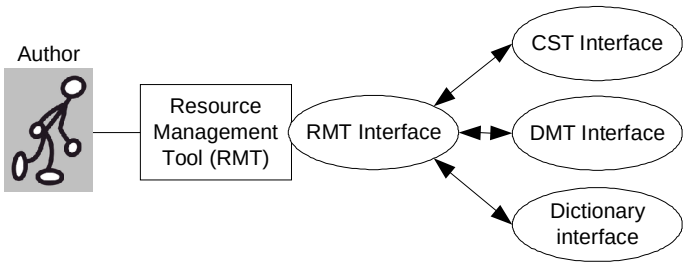


**Figure 20, RMT interaction**

As described in *paragraph 3.2 Use Case* the main tasks of the RMT are managing resources and searching in resources. Or more basic: reading resources, writing resources and determining a selection of the resources.
These basic functions are shown in Figure 21.

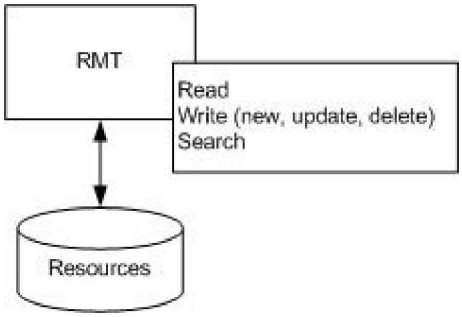


**Figure 21, the basic RMT functionality**

This definition of basic tasks for the RMT supports the partitioning of the RMT architecture in task-oriented components.

### 3.6.3 Analyzing RMT functionality

The Resource Management Tool has two main functionalities: searching resources and managing resource metadata. These two functionalities, the interaction with the other components (RMT interface) and the interaction with the author (Graphical User Interface) form the components for the RMT architecture as shown in Figure 22.

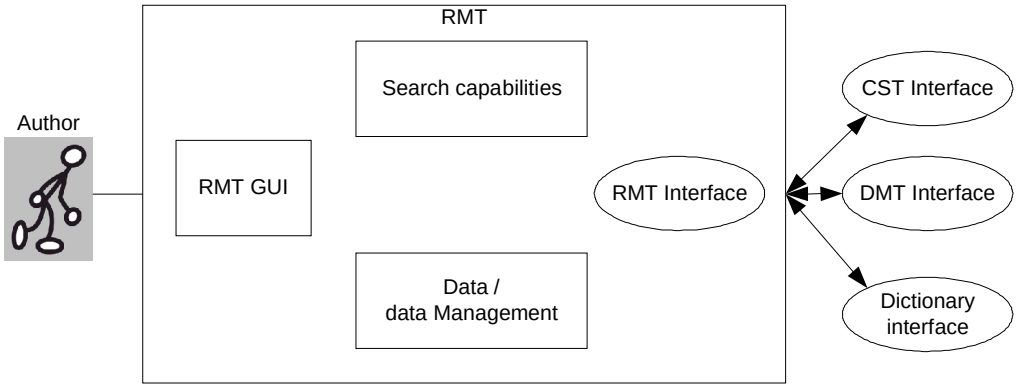


**Figure 22, main division of RMT in components**

The RMT GUI has two important functions: interacting with the author and managing the (presentation of) data in the user interface. For this reason we separate the RMT GUI into the program user interface and a 'presentation & feedback' component. For the other components we chose new names: 'Search' component and 'Resources' component. See Figure 23.

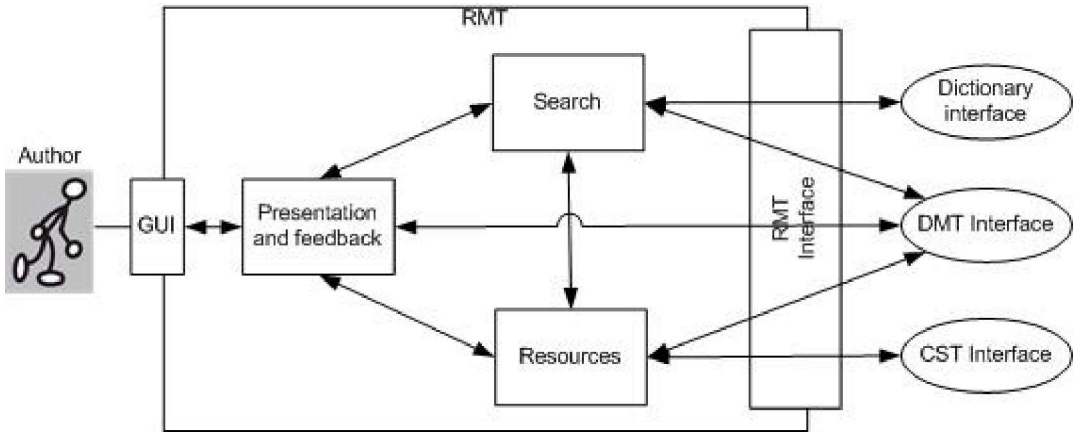


**Figure 23, RMT component communication**

Figure 23 shows the RMT components and the communication paths between those components and other systems. The communication paths through the RMT Interface are drawn to illustrate that the Search component uses DMT and Dictionary data to build a search. It also shows that the P&F component uses DMT information to represent relations in the GUI.

### 3.6.4 RMT components

The division of the RMT module into 3 components is large step in separating the tasks of the resource authoring process. The components have to be split up further to facilitate a better manageable structure.

3.6.4.1 Presentation & feedback component

The P&F is the component that supports the program user interface in gathering and formatting data that is displayed to the author. The Graphical User Interface of a program determines where a user clicks on the screen and what the user types, the actual handling of the system task that the author selects is done by the P&F.

The Presentation & Feedback component:

- Receives queries from the author via the GUI
- Reacts to selections or changes that the author makes in the GUI
- Receives notifications of changes in domains via the RMT interface

The P&F reacts on these inputs, dispatches tasks to other components and updates the user interface according to the changes in the system. To systematically handle these tasks the Presentation & Feedback component is split up into 3 subcomponents:

- Dispatch, receive tasks (queries, notifications) and pass these on to specific components.
- Presentation, update the GUI to reflect changes in the system or display a new view when the author changes selections.
- Feedback, display information about executed tasks, display errors and ask for confirmations from the author.

The division of the Presentation & Feedback component and its interaction with the other parts of the system:

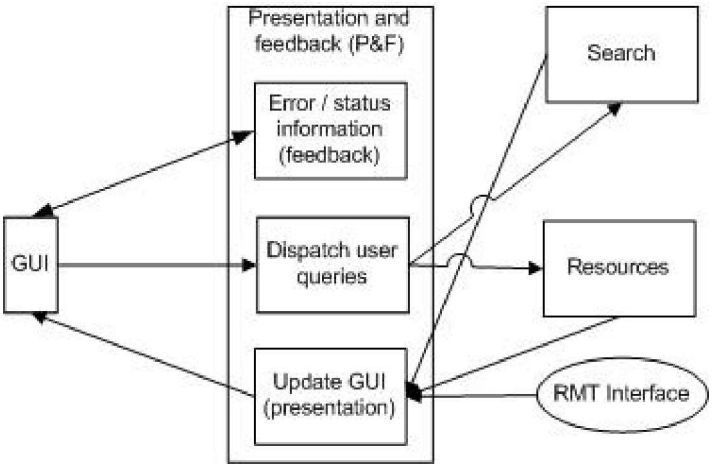


**Figure 24, P&F component and communication with other components**

3.6.4.2 Search component

The search component:

- Receives queries from the P&F component
- Extends a query using information from the dictionary (interface) and the DMT (interface).
- Requests resources matching the extended query from the Resources component
- Clusters resources matching the extended query into groups

Except for the clustering of resources into groups the tasks of the Search component are small or too much intertwined to split up. The Search component is split up into the following subcomponents:

- Search engine, handling and transformation of search queries
- Clustering, grouping of search results

The division of the Search component and the interaction of the Search component with the other parts of the system:

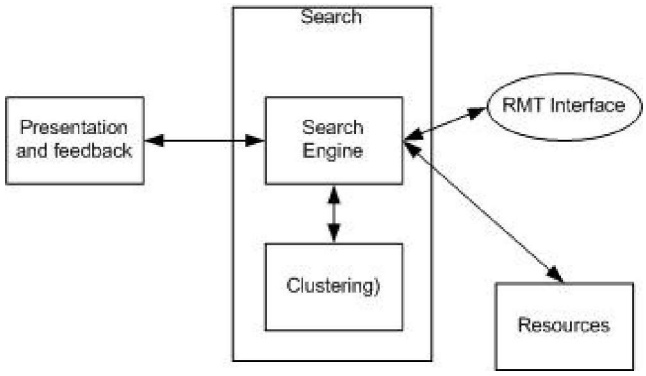


**Figure 25, Search component and communication with other components**

3. 6.4.3 Resources / Metadata component

The Resources component:

- Receives requests for metadata from
    1. The Search component, for resources matching search words
    2. The P&F component, for displaying data on the user interface and enabling the author to change the metadata
    3. The RMT Interface to pass thru to other authoring modules
- Receives updates for the resource pool from the RMT interface when relevant data changes in other authoring modules.
- Receives updates for the resource pool from the P&F component when the author modified a resource
- Reads/Writes its resource pool to and from disk

Except for the reading/writing of resources to disk the tasks of the Resources component are small or too much intertwined to split up. The Resources component is split up into the following subcomponents:

- Resource manager: manages resources that are loaded in the system.
- Reader/writer: reads and writes resources between memory and disk.

The division of the Resources and the interaction of the Resources component with the other parts of the system:

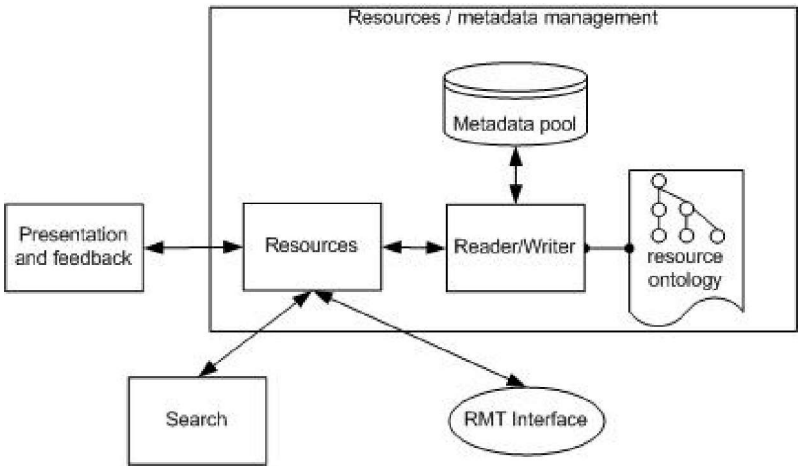


**Figure 26, Metadata component and communication with other components**

### 3.6.5 Final RMT architecture

This splitting of functionality and components of the RMT module leads to the following overview:

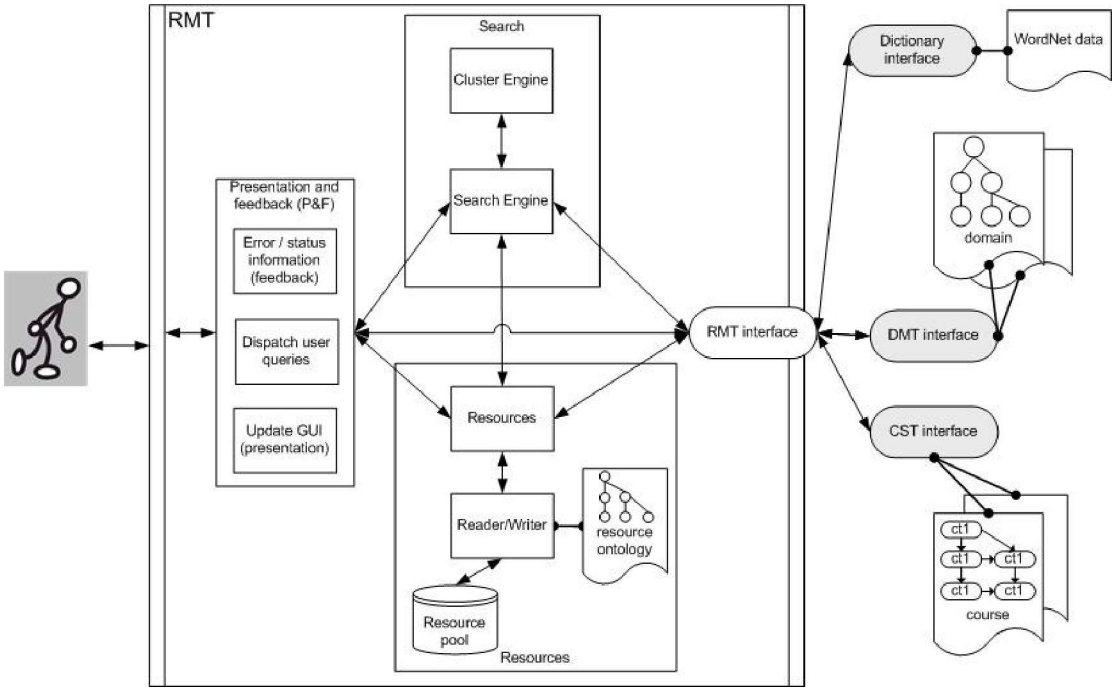


**Figure 27, Architecture OntoAIMS RMT authoring module**

### 3.6.6 Previous architecture design

Before the current architecture of the RMT was constructed other designs where created and rejected. The next figure illustrates an early design and why it was not suitable to build the RMT.

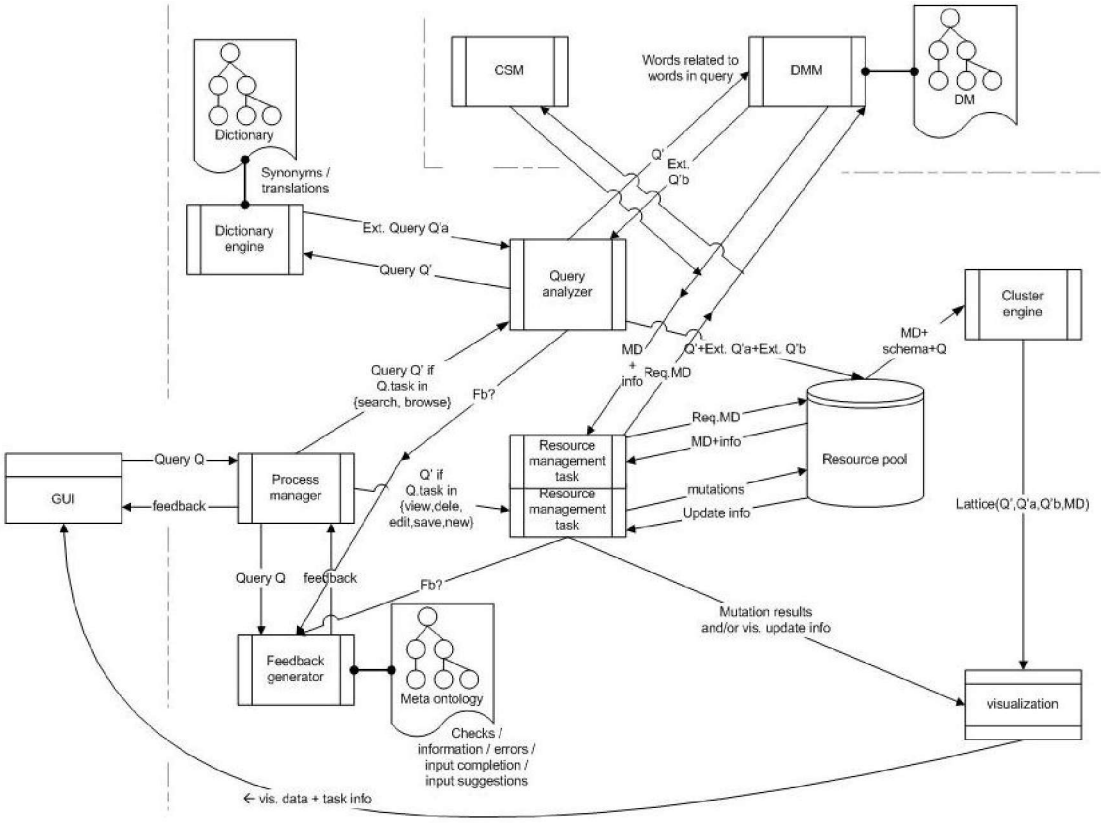


**Figure 28, old architecture 2**

Figure 28 shows an old architecture design. This model lacks uniformity, most subcomponents communicate with each other and with OntoAIMS components outside the RMT. This architecture lacks a dedicated point for 'outside communication' and has not much of a hierarchical structure, which makes it hard to define order in executing tasks.

The 'dictionary engine' is an internal component in this model. Since all OntoAIMS modules (RMT, DMM and CST) must be able to use the dictionary[10] it needs to be a separate OntoAIMS module.

It took a few tries and evaluations of usability to create a workable architecture for the Resource Management Tool. Just dividing the system in the main tasks the system has to provide did not create adequate results. Nor did splitting up the system in small sub tasks, even though those subtasks where reusable. In the end creating a single point of communication with the user, a single point of communication with the other OntoAIMS modules and a dispatching module for coordinating RMT tasks seems to make it work. (Figure 27)

[10] The dictionary provides synonyms to the RMT. But this task is not only useful for the RMT. The CST and DMT can also use it to look up synonyms or do spell checks, etc.

## 3.7 Using the architecture in search

With the finished architecture the tasks of the RMT can be projected as a flow of information in the RMT system. Figure 29 shows the task of searching in resources.

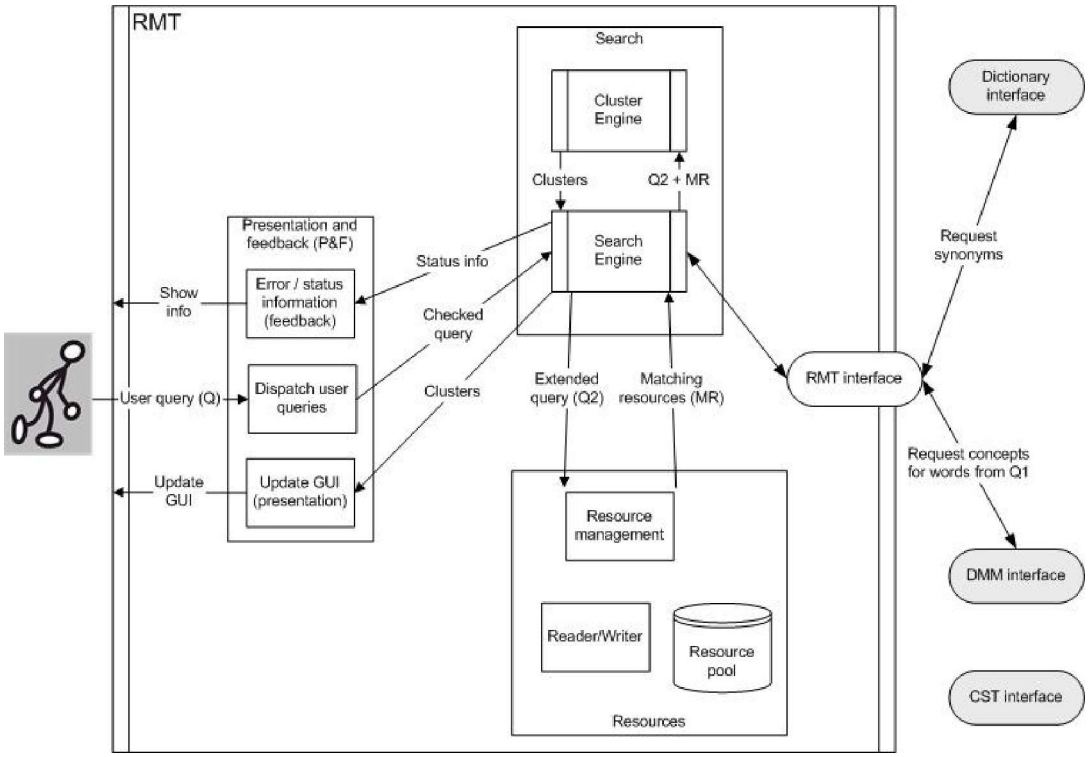


**Figure 29, path of a query through the architecture**

Figure 29 is an abstract of Figure 27. It shows only the flow of information used in searching. The arrows show the direction and 'type' of information that is exchanged between the architecture components. This flow of information is related to the steps of executing a search in the RMT system as specified in paragraph 3.6 Searching for resources. The following table relates the steps of paragraph 3.6 Searching for resources to Figure 29.

| Step # | Step action | Architecture component |
|---|---|---|
| 0 | Get query from author | Author (GUI) à Dispatch |
| 0 | Check syntax of query | Dispatch |
| 1 | Split query into words | Search engine |
| 2 | Filter query words | Search engine |
| 3 | Add concepts to query words | Search engine |
| 4 | Add synonyms to query words | Search engine |
| 5 | Get documents for extended query | Metadata |
| 6 | Make base clusters | Cluster engine |
| 7 | Make sub clusters | Cluster engine |
| (8) | Show clusters & information to user. | Presentation & feedback à GUI |

This analysis of traversal through the architecture (combined with the class structure of the next paragraph) is used in creating task sequences for the system (paragraph 3.9 Sequences).

## *3.8 Classes*

The architecture design of the RMT translates to UML class diagrams to specify the transition to actual components in the implementation of the authoring tool.

### 3.8.1 UML Class diagrams

[UML]: Classes are depicted as boxes with three sections, the top one indicates the name of the class, the middle one lists the attributes of the class, and the third one lists the methods. An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods). A class is a representation of an object and, in many ways; it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application.

### 3.8.2 Creating the class structure from the architecture

The class representation of the Resource Management Tool differs slightly[11] from the architecture in the additional specification of:

- Types for internal data storage and internal information interchange
- Detailed user interface components
- Utility classes for data manipulation

3.8.2.1 OntoAIMS main framework

The OntoAIMS authoring environment consists of four modules:

- Domain Modeling Tool (DMT)
- Course Sequencing Tool (CST)
- Resource Management Tool (RMT)
- Dictionary

These modules are separate but interact using each other's interface. The 'Module Handler' provides the access to the interfaces of the authoring modules. See Figure 30.

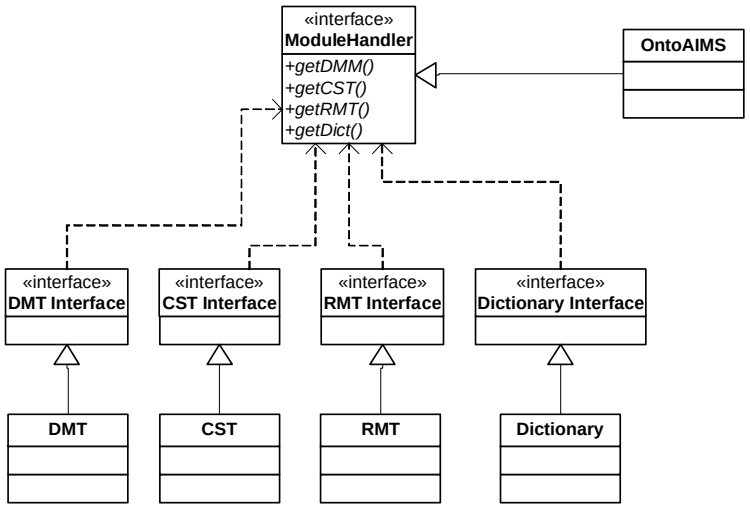


**Figure 30, OntoAIMS main class structure**

[11] The class representation has more details that are necessary or useful for the implementation of the application. Data types or specific user interface elements are too much detail for the architecture design.

3.8.2.2. RMT Class diagrams

The RMT class structure represents the RMT architecture with the RMT interface, 3 components (P&F, Search and Resources) and the user interface (GUI component). Added items are the RMT (main) class to link the interface to the internal components and the Types class for the specifications of resource types. The RMT class structure is shown in Figure 31.

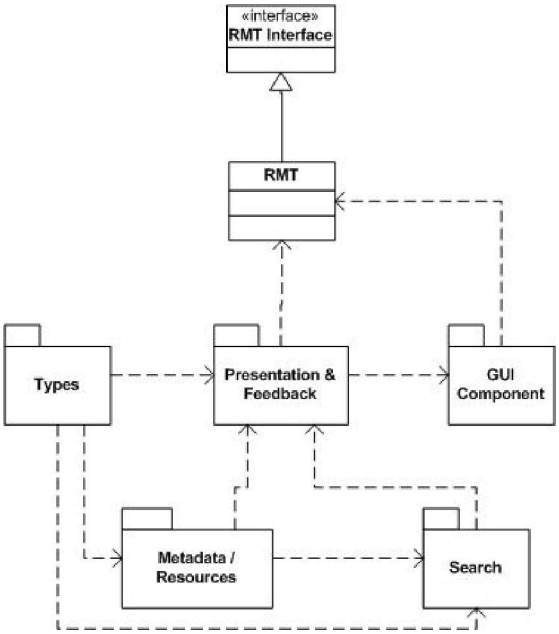


**Figure 31, RMT main class structure**

Within this structure the Types, P&F, GUI, Metadata and Search are so called packages; they contain more classes that are grouped together. The P&F, Metadata and Search groups are the components and divisions thereof mentioned in the RMT architecture design (3.6 Architecture).

Figure 32 shows the package of P&F classes. The P&F package contains three interfaces and one implementing class. The Dispatch class contains the functions that are triggered when the author selects data in the GUI or requests a change in the RMT data. The Feedback class contains the functions that present errors, warnings and status messages to the author. The Presentation class contains the functions that gather the data for and place data in the GUI. Functions from the Presentation class are executed to present the results of an action by the author or to update to changes in the DMT or CST.

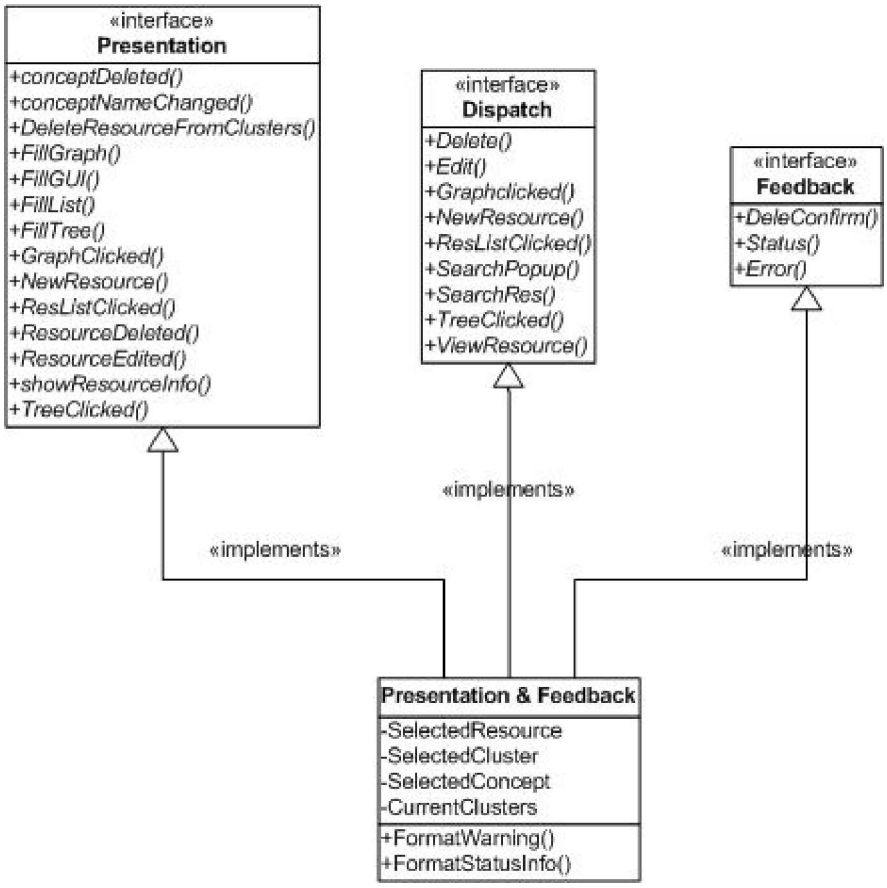


**Figure 32, RMT P&F class diagram**

Figure 32 shows the P&F package and its classes. The P&F classes perform the tasks of delegating operations and formatting data in the GUI.

The full Class diagram set is specified in Appendix 5: Classes.

## *3.9 Sequences*

Using the class diagrams (which form the blueprint for the authoring application) and the Use Case diagram the tasks of the Resource Management Tool are modeled into sequence diagrams to specify the main communication flows in the RMT.

### 3.9.1 OntoAIMS module communication

For the communications between the authoring modules – based on the required information from one module to another – a set of requests and messages is agreed upon (distilled from the architectures of the authoring applications). The communication of the resource management tool (RMT) with the other authoring modules is visualized in Figure 33:

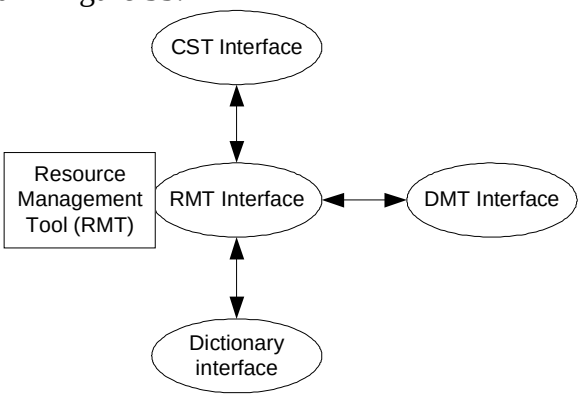


**Figure 33, Interface communication directions**

The communication between the modules is dividable in 2 types: messages and requests. The 'messages' are used to inform other modules that something has changed and no answer is required. For example: "resource X is deleted" or "concept Y changed its name to Z". Requests ask the other module for information and have an answer in a specified format. For example: "Give me the name for resource X" or "to which domain does concept Y belong".

Two exceptions are:

- There is no message to the other authoring modules at the creation of a new resource since the linking of a concept and a resource has different extra properties in the authoring modules. When creating a relation between a resource and a concept in the domain editor a property called 'knowledge propagation'[12] has to be set by the author. When this relation is created in the resource editor an author specified name has to be added to the relation description.

[12] Knowledge propagation is a method of specifying how much knowledge the user has about a subject or domain concept has when he/she reads the information on a related concept. The relation of knowledge propagation defines a number for the overlap in information / learning between two concepts. [Hubens]

- There is no notification to the other authoring modules when updating other aspects of the resource then the resource URI. Each authoring module is responsible for maintaining the metadata of the objects in its care and the other authoring modules request all other information about a resource they need by referencing the resource URI.

Example 1:

| | |
|---|---|
| java.lang.String | **GetResourceName**(java.lang.String ResourceID)<br>Answer the name of the resource identified by the String ResourceID. |

This request enables the DMT and the CST to ask the RMT for the name of a resource identified by the given URI.

Example 2:

| | |
|---|---|
| boolean | **ResourceExists**(java.lang.String ResourceID)<br>Answer whether a resource identified by the String ResourceID exists. |

This request enables the DMT and the CST to ask if the resource identified by the given URI exists in the resource pool (true or false).

The full set of communication requests and messages is specified in Appendix 7: Module communication. Appendix 7 and the examples are formatted using JavaDOC. See Appendix 6 for information on JavaDOC.

### 3.9.2 RMT sequence diagrams

The sequence diagrams for the tasks in the OntoAIMS Resource Management Tool are visualized by showing the involved classes and the requests/messages for the described task.

**Example 1:**
Create a new resource in the OntoAIMS resource pool. The OntoAIMS author has selected to input the metadata for a new resource in the GUI.

Sequence diagram:

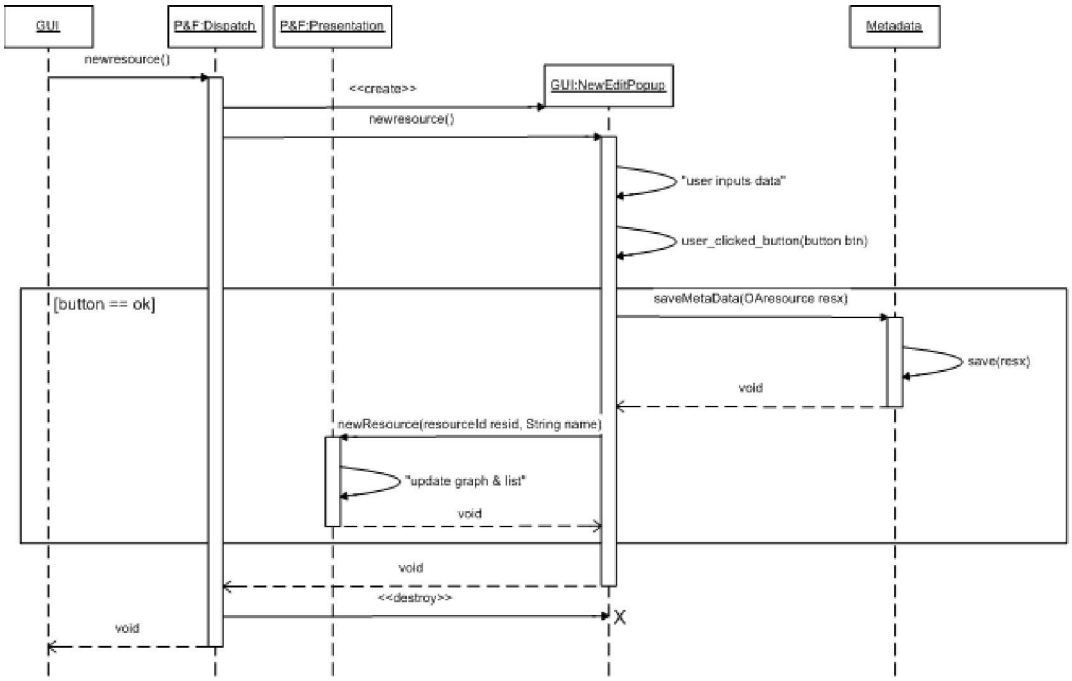


**Figure 34, sequence new resource**

Search sequence in words:
In the GUI the author selected to input a new resource either by selecting the 'new' option in the file menu or the 'new resource' option in a popup menu. The P&F dispatch starts an empty input dialog for the author to input metadata. The author presses the 'OK' button to commit the metadata to the system. The data is added to the internal data structure in the Metadata component and The P&F presentation component shows the created resource in the GUI.

Comment:
This sequence illustrates the creation of a new OntoAIMS resource. The author inputs the metadata and commits it to the system. The metadata is not written directly to disk. The author has to request the system to save data to disk or the system will ask the author if saving is required when the author closes the OntoAIMS application. As specified in the OntoAIMS internal communication, the DMT and CST are not informed of the creation of a new resource.

**Example 2:**
Get information on a resource. The CST module requires information on a specific resource. The CST requests information through the interface.

Sequence diagram:

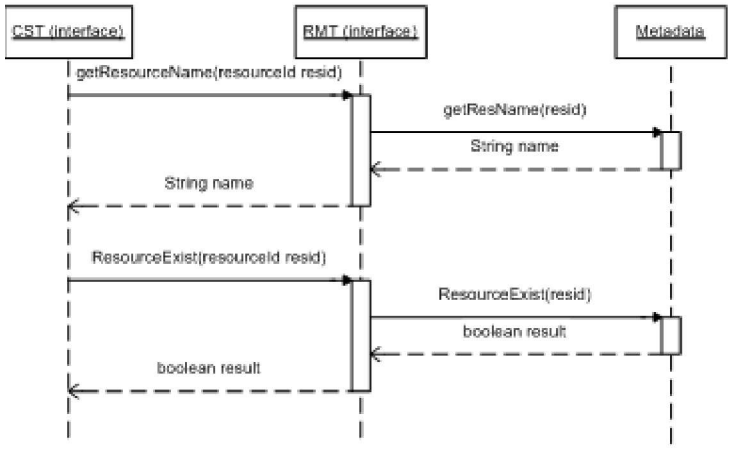


**Figure 35, sequence CST requests**

Search sequence in words:
(Top) The CST has a course that contains a reference to a resource. To display the resource name the CST requests that name from the RMT. The CST calls the RMT interface function “getResourceName” specifying the resource URI. The interface passes the request on to the Metadata component. The Metadata component returns the name and this name is passed on to the CST.

(Bottom)
The CST has a course that contains a reference to a resource. To check if this resource is still in the resource pool the CST requests a check from the RMT. The CST calls the RMT interface function “ResourceExists” specifying the resource URI. The interface passes the request on to the Metadata component. The Metadata component returns true or false and this result is passed on to the CST.

Comment:
This sequence illustrates the communication between OntoAIMS modules. It shows that the DMT and CST only specify the URI of a resource in their own metadata. All other metadata of a resource can be requested via the RMT interface.

## *3.10 RMT Graphical User Interface*

Based upon the graphical requirements for the RMT application, the visual aspects of AIMS and a generally defined[13] graphical framework the RMT user interface is designed. The sketches for the general framework are shown in Figure 36.

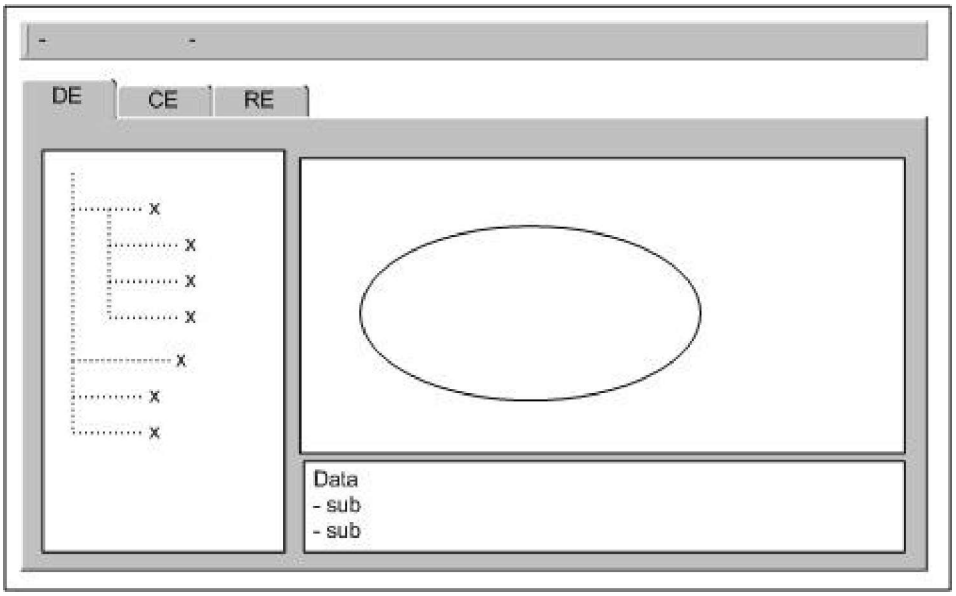


**Figure 36, OntoAIMS graphical framework sketch**

The Resource Modelling Tool is part of the OntoAIMS authoring environment, it is one of the three main components. These three components are organized by a *tab structure*, where each tab provides access to one of the tools of the authoring environment.

### 3.10.1 Main window

The browsing features of the AIMS viewer[14], the management features of the AIMS authoring module and the clustering of search results form the base for the RMT main user interface.

The main window (in the tabbed structure of the OntoAIMS GUI) of the RMT consists of five parts:

- A search form where the author can formulate a search request.
- A hierarchical navigation pane ('tree'), displaying the hierarchical structure of the clustering of search results.
- A list, which contains the (sub-) set of resources from the search results that belong to the currently selected cluster.
- A relationship navigation pane ('graph'), visualising the relations (depth=1) of the currently selected object (resource, concept or cluster).
- A description pane providing a description of the selected object (resource, concept or cluster).

[13] The general OntoAIMS graphical framework is a consensus of the ideas of the designers of the OntoAIMS components and work on the general framework started before the RMT design.
[14] The moving graph structure in AIMS is called the 'aqua browser'

The combination of these user interface elements allows the author to navigate the resources and related data (clusters & domain concepts). When a selection changes in any of the GUI elements the other elements change accordingly[15].

This defines the elements that build up the RMT GUI and comply with these requirements from paragraph 2.3 Requirements:

- Requirement 2.1: the RMT is part of the OntoAIMS system.
- Requirement 6.1: The RMT, DMT and CST share the same space in the OntoAIMS GUI.
- Requirement 6.3: The five elementary GUI parts mentioned in the previous list.

A representation of the RMT user interface design is shown in Figure 37. On the top are the menu structure and the tabs to switch between the authoring modules.
On the left are (top down) the search form, the cluster tree and the resource list. On the right are (top down) the relationship graph and the information / description pane.

In Figure 37 the *menu item for the resource editor* is activated and the menu items for the *Domain editor* and the *Course editor* are not. This complies with requirement 6.2: the menu for an OntoAIMS component is only active if that component has focus.

The arrows in Figure 37 show the order of data dependency.
If a cluster is selected in the tree then:

- The resource list is populated with the resources of the cluster
- The graph is redrawn with the selected cluster (selected object is placed in the centre and the related objects are placed around it)
- The information pane is filled with information on the selected object.

If a resource is selected in the resource list:

- The graph is redrawn with the selected resource (centred)
- The information pane is filled with information on the selected object.

If an object is selected in the graph:

- The information pane is filled with information on the selected object.
- Depending on the type of clicked resource items in resource list and cluster tree is selected[15].

These GUI data dependency specifications comply with requirement 6.4.
Requirement 6.4: selecting an object in the RMT GUI causes the other GUI elements to display relevant information if any is available to display in that GUI element.

[15] This selection will only be set when possible. Since the 'tree' only contains clusters, the resource list only contains resources and the graph can contain all three of the object types. If a selection in 'tree' or 'list' is not possible the selection will revert to none for that window part.

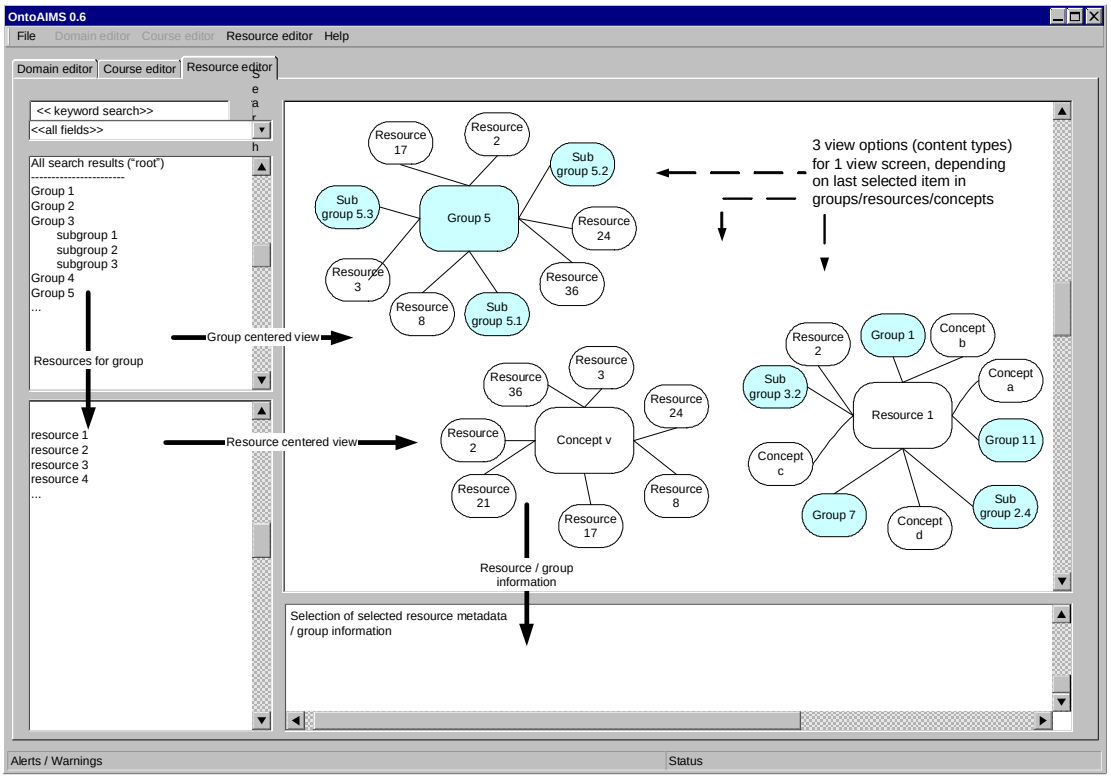


**Figure 37, RMT GUI design - main window**

Figure 37 shows the design for the main RMT user interface. On the top left the author can enter search words and start a search. Centre left the clusters for the search results. On the bottom left the author is shown the resources in the last selected cluster. Top right shows a graph with the last selected object (concept, resource or cluster) in the centre and related objects around it. Bottom left shows textual info on the last selected object. The last selected object in the GUI is always in the centre of the graph and if possible the same object is selected in the tree or list on the left.

The resource author is guided in managing resources and searching for resources in several ways:

- Dialogs explain choices
- Management actions can be initiated from several points in the user interface by using popup menus or the main 'resource editor' menu.
- Input fields are supported by extra information; the mouse-over-effect shows an explanation and an example.
- Some information might be suggested to the author if it can be retrieved by the system[16]
- Dialogs inform the user of inconsistencies or errors
- The author can switch between the OntoAIMS authoring modules (DMT, RMT and CST) without changing the state an authoring module is in.

[16] For instance: using the URI of a new resource the actual object on the Internet can be queried to get a 'last modified' date or document type.

This user interface guiding of the author comply with these requirements from paragraph 2.3 Requirements:

- Requirement 6.2: module menu is enabled when the module has focus.
- Requirement 6.8: the author is informed using dialogs.

For more compatibility with AIMS and more visual aid in browsing resources some additional functionality is added to the RMT user interface:

- History bar: searches and selections of objects through time[17] are visualised on a bar on the top of the RMT main screen. Clicking on an object in the bar selects the object in the graph and that changes the selections in the other graphical components accordingly.
- Relation highlighting:
    - When the author moves his/her mouse over an object in the graph a subset of resources in the resource list are highlighted to show that they are connected to the object under the mouse pointer.
    - When the author moves his/her mouse over an object in the graph the information of that object is shown in the information pane.
    - When the author moves his/her mouse over an object in the resource list a subset of objects in the graph are highlighted to show that they are connected to the object under the mouse pointer

These visual aids offer the author:

- An overview of earlier actions in the system.
- The ability to go back to a view on the data he / she selected earlier.
- More information on a data item in the GUI without having to select it.
- More insight in relations between data items in the GUI.

[17] A maximum of about 10 objects is shown in the history bar, depending on the width of the screen

### 3.10.2 Edit dialog

In paragraph "3.3.3 OntoAIMS Resource metadata" the metadata for a resource was determined. The resource metadata consists of several field grouped into categories. These groupings are also useful for dividing the editing dialog into manageable parts. This division is accomplished using tabs.
Each editable field has a description and an example of what data is expected.

For consistency the same dialog is used for creating a resource, editing a resource and viewing resource metadata. Depending on the 'mode' of the dialog the fields are or are not editable.

A representation of the metadata edit dialog design is shown in Figure 38.

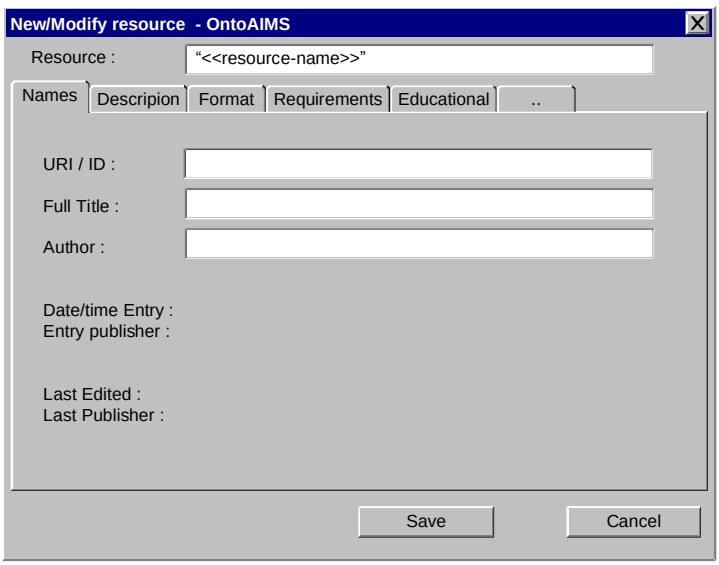


**Figure 38, RMT GUI design – metadata edit dialog**

Figure 38 shows the dialog where the author can insert, change and read the metadata of a resource. This dialog is used in three modes:

- View resource: The author can view the information on each tab but the author cannot change any data. Only a 'close' button is available on the bottom of the dialog.
- New resource: The dialog has empty data fields. The author can type data in the resource metadata fields.
- Edit resource: The metadata of a resource, selected by the author, is loaded in the dialog. The author can change the resource metadata.

# 4. Implementation

The developed plans described in the previous chapters are implemented into an actual application. These plans are:

- The use case and architecture models,
- The OntoAIMS module communication protocol,
- The class structure and sequence diagrams and
- The resource search algorithm

The application is built using the development environment specified in the next paragraph.

## *4.1 Development environment*

### 4.1.1 Application

The authoring environment of OntoAIMS is constructed as a standalone application in contrary to AIMS that is a web based authoring application. Instructors usually work at the same computer and have no need in downloading the environment from a server every time they use the tool to start work on domains/courses/resources.

The OntoAIMS authoring environment consists of the application, java libraries, the OntoAIMS data sets and the WordNet dictionary. This package makes it a large download. Downloading would be very time consuming and it would not be possible to work if the Internet connection is slow or unavailable.

In the planning for the development of the DMT it was already decided to make the OntoAIMS authoring environment a stand-alone tool.

Just like for AIMS Java was selected for de development language. Java is operating system independent, offers libraries and plug-ins to add functionality and Java is a popular language in web-related development.

The OntoAIMS Java code is edited using JBuilder 9. JBuilder provides time saving options like code completion, debugging, a graphic designer, a JavaDOC wizard and more.

### 4.1.2 Documentation

The documentation for the application is entered together with the code in Java. Using the JavaDOC coding tool – developed by Sun - special comment sections are inserted in the code. The JavaDOC tool parses the code and generates a structured HTML file for each Java class describing the commented sections, function headers and Java package order.

JavaDOC is described in Appendix 6: JavaDOC specification.

### 4.1.3 Libraries

Specific Java libraries have been chosen to assist with some tasks in the RMT application. The libraries provide generic access to data storage, language manipulation or visualisations.

4.1.3.1 Jena

The Jena library [Jena] is used for reading and writing resources to and from disc in RDF format. Jena reads / writes between the RDF[18] file on disc and its own memory model of the data.

Product description according to [Jena]:
*Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Programme.*
*The Jena Framework includes:*

- *A RDF API*
- *Reading and writing RDF in RDF/XML, N3 and N-Triples*
- *An OWL API*
- *In-memory and persistent storage*
- *RDQL – a query language for RDF*

Jena uses its own memory model to read from and write to RDF files. This memory model needed to be mapped to the internal data structure of the RMT. The RMT Metadata reader/writer component instructs Jena to read the RDF resource file. Then, using the RMT resource ontology, the reader/writer maps the resulting Jena data structure to the (in-memory) resource data. The other way around the reader/writer converts the RMT resources to a Jena model, which Jena then writes to RDF.

Besides the fact that the DMT also uses the Jena library for reading/writing to file these other XML reading/writing tools were considered for this project:

- Xerces Java Parser, by Apache. The Xerces parser parses XML data but not specifically RDF/XML. [Xerces]
- Saxon. The Saxon parser focuses mainly on XSLT and Xpath, which are specific uses of XML, but not on RDF. [Saxon]
- BorlandXML. The BorlandXML parser relies on being used in an Enterprise JavaBeans application. The OntoAIMS tool uses no EJB's. [BorlandXML]

Several plain XML parsers can be equipped with an RDF filter. Or an RDF filter can be used on the parsed XML data. This adds an extra, unnecessary, step to the reading/writing process.

[18] See also paragraph "4.2.1 RDF".

#### 4.1.3.2 JGraph

The Java Graph Visualization Library [JGraph] is used for displaying and visually navigating relations between objects (resources, concepts and clusters). The graph visualizes part of the relational structure between the objects in the system. The 'view port' on the relational structure that the graph displays changes with the selections the author makes.

Product description according to [JGraph]:
*JGraph (pronounced "jay-graph") is a robust and complete graph component that combines the power of open source development with the stability of commercial support.*

The JGraph library provides the programmer with a framework to build his/her own visualisation of data. JGraph offers the programmer a basic graph memory model, a canvas to draw on and 'cells' to place on that canvas. Out of the box these cells are shapeless, do not react to user actions and offer no information. All aspects of the JGraph cells have to be built by the programmer.

To visualize the RMT data in the JGraph several adaptations were written:

- A specific instance of the JGraph model
- Functions to create graph cells that contain OntoAIMS data objects and the visual aspects thereof.
- Functions to move and arrange cells on the canvas.
- Listeners to react on user actions
- Functions to (temporarily) change a cell's appearance. This is done to visualize selection or data relevance.

The DMT also uses the JGraph library for visualizing a part of the data set and relations. Besides JGraph these other visualization tools were considered for this project:

- The AquaBrowser by Medialab [AquaBrowser]. An older version of the AquaBrowser was used in the AIMS system. The AIMS system is web based and not a stand-alone application. AquaBrowser uses web libraries in its operations and is not just a visualisation engine. The software is not open source or free.
- Cluster Map by Aduna [ClusterMap]. At the development stage of the RMT Aduna had discontinued the Cluster Map library. Later on a new version of Cluster Map came out. Although it looks good and does the visualization necessary for the RMT, Cluster Map is not open source or free.

In writing the RMT implementation of the JGraph framework several problems arose:

- The RMT application would sometimes crash because JGraph was still drawing the effects of a selection or a mouse movement when the user was already doing something in another part of the GUI. Extra safeguards had to be built.
- Implementing part of the tutorials and adapting the data structure to that of the RMT made the first implementation of the graph. This model did not support the reuse of cells upon creating a new set of objects in the graph. The creation of cells and the memory model were rewritten.
- Partly because of a bug in the version of JGraph used en partly because the example code in the tutorial did not support it drawing selection edges around cells did not always work. The creation of cells and the memory model were rewritten.

#### 4.1.3.3 JWNL

The Java WordNet Library [JWNL] is used to access WordNet as a thesaurus. Using WordNet search queries can be extended with the synonyms of words to get more (equivalent or similar) search results.

Product description according to [JWNL]:
*JWNL is an API for accessing WordNet-style relational dictionaries. It also provides functionality beyond data access, such as relationship discovery and morphological processing.*

To easily use the JWNL library a wrapper[19] was written. The JWNL requires the use of specific settings and linked lists in order to retrieve WordNet data. In this wrapper a series of functions was made to retrieve lists of words from the JWNL API. The wrapper sets all the API specific settings, does the error checks and creates clean lists of results.

Besides JWNL these other WordNet tools were considered for this project:

- JWordNet [JWordNet]. The JWordNet library is much like the JWNL but the author has discontinued work on the library. The JWordNet author continued his work in Python.
- WordNet JNI Java Native Support [WNJN]. The WNJN package uses C methods to access the WordNet data structure. An additional C++ compiler and some other libraries are necessary to use this API.

[19] Software that accompanies other software for the purposes of improving convenience, compatibility or security.

In writing the JWNL wrapper some problems arose:

- When a requested word does not exist or has no synonyms in WordNet, the JWNL produces either an error or an empty value. These results would sometimes crash the application and had to be rewritten to empty list results.
- JWNL needs a configuration file to locate the WordNet data on the computer. In this file the path to the WordNet data is hard coded. Since a set of WordNet data will be distributed with the OntoAIMS application this path needs to refer to where the OntoAIMS application is on the PC and not to any other fixed position. An adaptation was created to read the configuration file from disk into a virtual file, alter the path to the WordNet data and pass the virtual file to the JWNL.

## *4.2 Resource representation*

The resource pool is saved in the Learning Objects Metadata (LOM) format (with some extensions as described in paragraph “3.3.3 OntoAIMS Resource metadata”) using the Resource Description Framework (RDF) format. The resource pool file can be read using a RDF interpreter for LOM or for Dublin Core (DC) without any conversion.

The OntoAIMS resource pool is stored to disk in a RDF model, which is written in XML syntax in a plain text file.

### 4.2.1 RDF

RDF (Resource Description Framework) [RDF] is a model for storing graphs of information. RDF is used to describe metadata for all kinds of resources (such as a person, a song, a web page or a bookmark) In RDF the nodes can be of two very general types, resources and properties. A property is a specific aspect, characteristic, attribute, or relation used to describe a resource. A property is always a combination of a property name (type) and a literal (value) for that type. A literal is an actual value, such as the name 'Sandra' or the number '7'.A resource is a representative of something, more like an object in a programming language. For example, a person would be a resource, but the name of the person would be a literal. A resource is always named by a URI plus an optional anchor id.

The value of the URI is irrelevant to RDF since it's just used as an identifier. But preferably we use the URI identifying the resource on the web or in a library or the URN of a person, etc. as the URI for the resource.

Data in RDF is modelled in the form of a set of statements. A statement about a resource is a triple that contains some information about that resource. A statement consists of:

- The subject: the resource about which the statement is made.
- The predicate: a certain attribute or characteristic of the subject.
- The object: the value of the attribute or characteristic mentioned by the predicate. This value can be another resource or a literal (data about which no further statements can be made, e.g. a string, an integer, a date or any other primitive data type defined in XML.)

An example of a RDF statement in N3[20] is:

```
<http://www-130.ibm.com/developerworks/rational/library/1147.html>
<rmt:Resourcedate>2005.01.27 21:53
```

This statement specifies that the resource identified by http://www-130.ibm.com/developerworks/rational/library/1147.html was inserted in the resource pool on the date of January 27th 2005.

### 4.2.2 Namespaces

A single XML document contains elements and attributes (also referred to as a "mark-up vocabulary") that are defined for and used by all kinds of software tools. Applications (software tools) need to be able to recognize the tags and attributes, which they are designed to process. To prevent "collisions", occurring when mark-up can belong to more then one definition standard, the mechanism of XML namespaces identifies the set of tags where a tag[21] is specified.

Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a namespace prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace. The combination of the universally managed URI namespace and the document's own namespace produces identifiers that are universally unique. URI references, which identify namespaces, are considered identical when they are exactly the same character-for-character. Note that URI references, which are not identical in this sense, may in fact be functionally equivalent.

A namespace is declared using a family of reserved attributes. Such an attribute's name must either be xmlns or have xmlns: as a prefix.

[20] The N3 notation is a triple of a URI (subject), a predicate and an object. Mostly the RDF data is saved in XML. The XML notation for the example is: <rdf:Description rdf:about="http://www-130.ibm.com/developerworks/rational /library/1147.html"> <rmt:Resourcedate>2005.01.27 21:53</rmt:Resourcedate> </rdf:Description>.

[21] A tag in RDF is called “predicate” or “property name” and in paragraph “3.3.3 OntoAIMS Resource metadata” is referred to as “descriptor”.

The following namespaces are used for describing OntoAIMS resources:

| | |
|---|---|
| A | `xmlns="http://purl.org/rss/1.0/"` |
| B | `xmlns:rmt="http://localhost/ontoaims/rmt/schemas/1.0/"` |
| C | `xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"` |
| D | `xmlns:dc="http://purl.org/dc/elements/1.1/"` |
| E | `xmlns:lom="http://kmr.nada.kth.se/el/ims/schemas/lom-general/"` |

The name spaces describe the following:

| | |
|---|---|
| A | General XML syntax |
| B | Own additions to LOM syntax |
| C | RDF definitions |
| D | Dublin core syntax, tags that are both in LOM and in DC are written in DC |
| E | LOM general definition, LOM has more than one name space but the general one is used most |

For example: the tag “DC:Title” is called ‘Title’, is part of the definition of Dublin Core (DC) and is specified at location “`http://purl.org/dc/elements/1.1/`”.

### 4.2.3 RMT resource pool

The RMT resources are written to and read from a single RDF / XML file: the RMT resource pool. The RMT resource pool is read/written using the Jena library. An example of a RMT resource is shown in Appendix 2: RMT resource example.
The requirements, keywords and relations for a resource are separate RDF objects because they are compound types. The order in which RDF objects occur in the resource pool are irrelevant since the data from the resource pool is interpreted by Jena by matching up identifiers (URI’s). A visualization of the resource pool is shown in Figure 39.

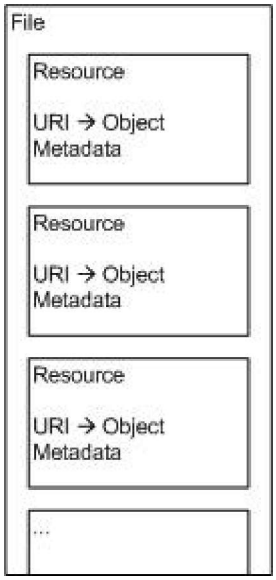


**Figure 39, RMT resource pool file**

## *4.3 RMT application*

With the specifications from the previous chapters (resource pool, architecture, classes, scenario's, architecture, classes, etc) the RMT application was built. This paragraph specifies what parts of the RMT application were built (paragraph 4.3.1) and what the resulting application looks like (paragraphs 4.3.2 and 4.3.3).

### 4.3.1. Realised parts of the RMT

Building the RMT entailed:

- Installing the libraries into the project:
  - JWNL
    - Writing functions to retrieve WordNet data using JWNL (JWNL wrapper)
    - Writing a dynamic configuration file for JWNL
    - Writing a 'Dictionary interface' for public access to functions on JWNL
  - Jena
    - Creating a model to map Jena memory structure to OntoAIMS resource memory model. (Resource ontology)
  - JGraph
    - Creating an adaptation of JGraph for relationship graph in RMT GUI
    - Creating an adaptation of JGraph for History bar in RMT GUI

- Building the RMT components from the architecture:
  - Metadata module
    - Metadata component. Creating metadata functions and an internal resource structure
    - RDF reader/writer (Jena wrapper)
  - Presentation and Feedback module
    - Dispatch component.
    - Presentation component
    - Feedback component
    - Graphical user interface components
      - Main user interface
      - Search user interface
      - Resource edit/view dialog
      - Resource relation edit dialog
      - Resource requirement edit dialog
      - Error dialog
      - Information dialog
  - RMT main class
  - Search component
  - Data types

- Creating the resource pool:
  - Creating the resource pool structure
  - Creating demo resources

### 4.3.2 Main window

Figure 40 shows the main screen of the RMT application. The view is of the search results for the query “boat”. The query “boat” was extended by the Search component to the query “boat” + “gravy boat” + “gravy holder” + “sauceboat” (synonyms from WordNet) and the query was matched to the concepts “Vehicles: Rowboat” + “Vehicles: Sailboat” + “Vehicles: Boat”. With this new query 8 resources are found in the resource pool of 111 resources.

The search results are clustered according to belonging to the same domain, connected to the same concept, having the same type, having the same source, etc.

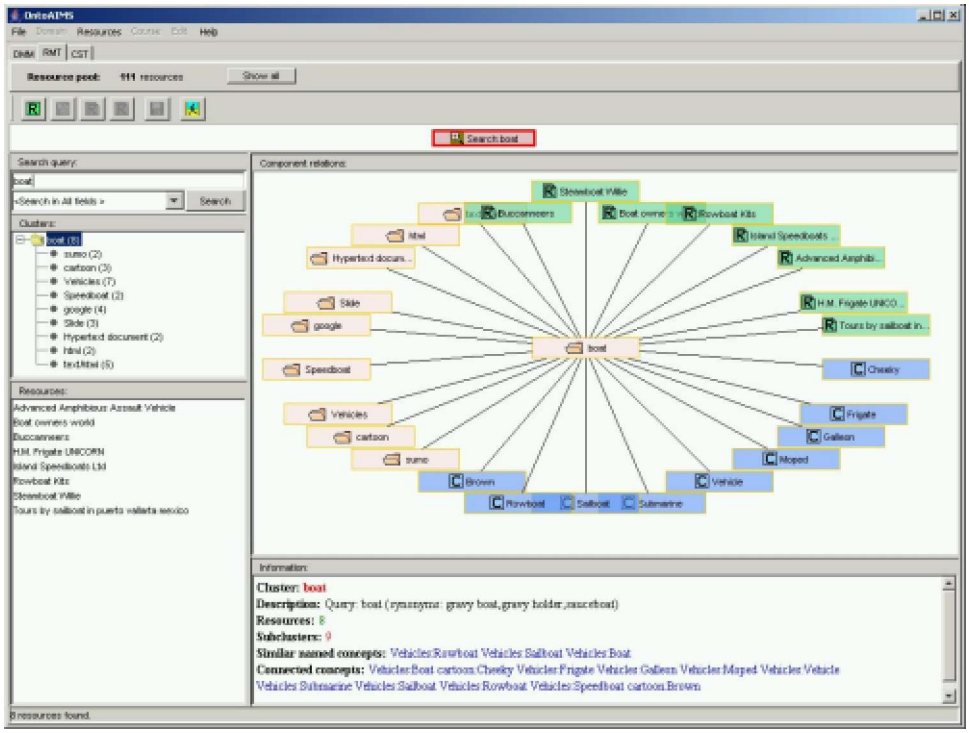


**Figure 40, OntoAIMS RMT main screen - after search**

Figure 40 is a screen capture of the RMT application.

Top down:
Information bar reading 111 resources in the resource pool
Button bar with buttons “new resource” and “switch user” enabled
History bar with object “Search: boat”

| Left: | Right: |
|---|---|
| Search form with query “boat”<br>Cluster tree with clusters for search “boat”<br>Resource list for cluster “boat” | Relationship graph with cluster “boat” in centre<br>Information pane with data about cluster “boat” |

Bottom:
Status bar

### 4.3.3 Edit dialog

Figure 41 shows the new/edit/view screen of the RMT application. The view is of the creation of a new resource called “Van Dale Taalweb On-line woordenboek”.

For the creation / editing / viewing of a resource the following items are displayed:

| **Names** | **Description** | **Format** | **Requirements** | **Educational** |
|---|---|---|---|---|
| URI | Description | Format | Requirements | Interactivity type |
| Source | Language | Size | Relations | Learning type |
| Author | Coverage | Logical size | | Interactivity level |
| Date | Keywords | | | Semantic density |
| Entry | | | | Difficulty |
| Last modification | | | | Context |
| First editor | | | | |
| Last editor | | | | |

The words on top line of this table are the labels that are on the tabs in Figure 41 and are the groups specified in paragraph “3.3.3 OntoAIMS Resource metadata”.
The name of each of these groups is the name of a collection of metadata descriptors in the LOM standard. The metadata elements that are in the RMT metadata set that are not specified in LOM are placed in the group with the name that best suites for describing the metadata element.

The metadata elements that are used in RMT resources are specified in paragraph 3.3.3 OntoAIMS Resource metadata and in Appendix 1: metadata tags.

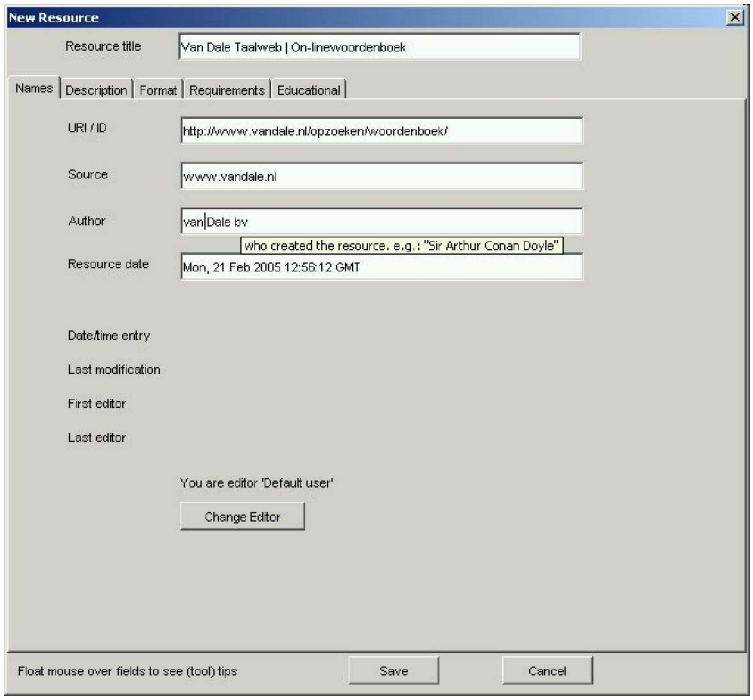


**Figure 41, RMT new resource dialog – first tab**

Figure 41 shows the RMT New/Edit/View dialog (screen). In the dialog an author is entering the metadata aspects of the resource he/she wants to add to the resource pool. In this view the author is typing in the name of the author of the resource. A 'tool tip' shows the author an explanation of the data that is expected in the field and an example.

A more elaborate overview of the screens and dialogs of the OntoAIMS RMT application can be found in Appendix 9: OntoAIMS screenshots.

# 5. Conclusions and evaluation

## *5.1 General conclusions*

The goal of this project was to provide requirements, architectural design and implementation of a Resource Management Tool (RMT). The RMT is part of the OntoAIMS system/project. The OntoAIMS system consists of a Course Sequencing Tool (CST), a Domain Management Tool (DMT) and a Resource Management Tool (RMT). The OntoAIMS project is an elaboration on the AIMS project, which was built and refined over several years of research.

The AIMS system stored its data in a propriety format. The OntoAIMS system uses open web standards for metadata annotation to store its information. The AIMS system manages datasets that are composed of one domain, some resources and courses on that domain and those resources. Resources could not be reused across domains. The OntoAIMS tools each manage their own data set (resources, domains and courses). All OntoAIMS resources can be linked to all domain concepts and can be used in all courses. The RMT, DMT and CST have a similar look and feel. The tools work together, share information and support the author.

For the DMT and the CST there already had been extensive research into the usable specifications for open web standards for metadata representation. The research on a metadata format for the RMT is limited to formats that can be used alongside the OWL specification (for DMT) and the SCORM specification (for CST). Much attention was given to:

- Metadata items to describe a resource,
- Metadata items to describe the communication between the authoring modules,
- Metadata items to support the display of resource relations and
- Metadata items to support the searching for resources.

As a basis for the requirements for the RMT the AIMS system was evaluated. The desired OntoAIMS module communication was taken into account and the specifications of the DMT were inspected. The DMT specifications were studied to create a similar look and feel of the modules and to determine the necessary communication.

Splitting tasks and responsibilities within the RMT into subsystems defined most of the RMT architecture. The tasks and responsibilities of the system are based on a Use Case evaluation of the operations the author has to be able to perform on the RMT.

After creating the specification the OntoAIMS RMT was implemented. The RMT was built using the architectural design, the class specification and the sequence diagram specifications.

The RMT offers multiple views on the available data in the resource pool:

- It shows relations to objects from the DMT / CST,
- It makes searching for resources easier by:
  - Browsing resource relations,
  - Extending search queries to find more results and
  - Grouping search results.

The RMT allows the author to create more descriptive resources, stored in open web standards for metadata specification.

## *5.2 Evaluation of the scenario's*

In paragraph "*1.5 OntoAIMS resource scenarios*" three scenarios describe the actions that the author would want to perform on the RMT application. After designing and building the RMT those three scenarios are evaluated. Each scenario is mapped to actions that yield the proposed result in the OntoAIMS system.

Scenario 1: Creating resources (short version)
*Thomas wants to prepare a course on the subject of Thermodynamics. He has already selected books and readers. He asked 2 colleagues to support the students. Thomas has to enter all these and more resources into the OntoAIMS system. With these resources and a domain ontology (built in the DMT) the course can be created. Thomas starts the OntoAIMS authoring environment and he goes to the resource editing part of the system. In the resource authoring section he selects that he wants to enter a new resource into the system. For the resource he enters into the system he has to have several description properties. The system will give Thomas a set of 'input boxes' to describe the resource. Thomas repeats the procedure of entering metadata for all the resources he wants to add to the system. Thomas tells the system to save the resources (permanently to disk) and closes the authoring environment.*

Steps for scenario 1
The following steps in the RMT represent scenario 1:

- Step 1: the author starts the OntoAIMS application/system.
- Step 2: the author clicks on the RMT tab to select the Resource tool.
- Step 3: the author selects to create a new resource:
  - Click with right mouse button on GUI and select from popup menu 'new resource'. And.
  - Select 'new resource' from RMT menu on the top of the GUI.
- Step 4: the author flips thru the tabs of grouped input fields, types in data and/or selects the appropriate data from pull-down lists.
- Step 5: the author clicks on the save button to commit the resource data to the internal memory structure.
- Step 6: repeat steps 3-5 for all resources
- Step 7: Save resources and close OntoAIMS

- Click on button ‘Save all’; Select menu ‘File’, item ‘Exit’. Or
- Select menu file, item ‘Exit’. The system knows that the data has changed and asks the author if he/she wants the data saved to disk.

Remarks for scenario 1
The steps describe a repeating action of opening the edit dialog, inserting data and saving data. This scenario is possible in the RMT.

Scenario 2: Looking through resources (short version)
Vincent wants to look up the coverage on the topic of conversion of energy from one form to another. Vincent enters some search words into the system. The OntoAIMS system interprets the query and gives feedback to the author if necessary (if there is an error in the query or the query yields no results). The author is shown a formatted result set for the query. With the extended query the system finds 15 resources in the resource pool. These found resources are grouped into sets. Vincent now goes through the search results by clicking on resources, viewing information and browsing the relations between resources, groupings and concepts. When he is done viewing resources and relations Vincent closes the authoring environment.

Steps for scenario 2
The following steps in the RMT represent scenario 2:

- Step 1: the author starts the OntoAIMS application/system.
- Step 2: the author clicks on the RMT tab to select the Resource tool.
- Step 3: the author starts a search:
  - Fill in search words. And.
  - Click the search button.
- Step 4: the author clicks on items from the search results to view relations and short descriptions.
- Step 5: Select menu ‘File’, item ‘Exit’

Remarks for scenario 2
These steps describe the actions of searching and browsing results. The scenario does not take into account what steps are necessary to gather and format the search results. This scenario is possible in the RMT.

Scenario 3: Updating resources (short version)
Charlie wants to teach a similar course on thermodynamics. Some information might not be accurate any more. Charlie opens the OntoAIMS authoring environment and finds out which resources are used in the course. Using the resource authoring section Charlie looks up the books he needs and cross-references them with the University Library. He deletes resources on books that are no longer in circulation and he adds the new books. When Charlie deletes the old resources the system tells Charlie that those resources where attached to courses and domains, he will link the new books to those concepts and courses later on. For documents that changed or were moved Charlie opens each resource, one at a time. The system shows him the resource metadata in an edit dialog. Charlie updates the information and selects to save the resource. After modifying the last resource Charlie tells the system to save the resources and closes the authoring environment.

Steps for scenario 3
Generic steps of scenario 3:

- Step 1: the author starts the OntoAIMS application/system.
- Step 2: the author clicks on the RMT tab to select the Resource tool.

The following steps in the RMT represent delete part scenario 3:

- Step 3: the author starts a search for a resource on his list:
  - Fill in the resource title. And.
  - Click the search button.
- Step 4: the author deletes the resource:
  - Click with the right mouse button on the resource for the book and select from popup menu 'delete resource'
  - The system asks the author if he is sure. Click 'yes'.
- Step 5: repeat step 3-4 for other outdated resources.

For adding new resources see scenario 1.

The following steps in the RMT represent edit resource part scenario 3:

- Step 6: the author starts a search for a resource on his list:
  - Fill in the resource title. And.
  - Click the search button.
- Step 7: the author edits the resource:
  - Click with the right mouse button on the resource for the book and select from popup menu 'edit resource'
  - The author flips thru the tabs of grouped input fields, types in data and/or selects the appropriate data from pull-down lists.
  - Click on 'save' button.
- Step 8: repeat step 6-7 for other changed resources.

Generic steps of scenario 3:

- Step 9: Save resources and close OntoAIMS
  - Click on button 'Save all'; Select menu 'File', item 'Exit'. Or
  - Select menu file, item 'Exit'. The system knows that the data has changed and asks the author if he/she wants the data saved to disk.

Remarks for scenario 3
These steps describe the editing and deleting of resources. The scenario does not take into account what steps are necessary to gather and format the search results.
This scenario is possible in the RMT.

## 5.3 Evaluation of the research questions

In paragraph *"2.1 Research questions"* and paragraph *"2.2 Project goals"* the research questions and project goals for the RMT are specified. After designing and building the RMT those research questions[22] are evaluated.

Research question 1: How to reduce the complexity of searching for resources for the author?

- 1.1 How to use the ontology (structure) to aid in searching resources.
- 1.2 How to visualize part of the (overall) OntoAIMS ontology, i.e. relations between OntoAIMS data objects (resource, concept, domain and course) and to show resources in context.
- 1.3 How to combine keyword search and ontological (relationship) browsing.
- 1.4 How to visualise common properties between resources in search results.

Evaluation of research question 1:
To reduce the complexity of searching for resources for the author (RQ1) these parts of the RMT were designed:

- The ontological structure (RQ1.1) supports the author by:
  - Showing (mutual) relations of resources to domain concepts and courses.
  - Grouping search results by common links to concepts, courses, domains and other resources.
  - Using concept relations to find more search results.
- The Visualizing of the OntoAIMS ontology around resources (RQ1.2) is done by showing a graph view to the author. This graph view shows a resource from the resource pool and the ontology items that are directly linked to that resource.
- Keyword search and relational browsing (RQ1.3) are a large part of the main GUI. The search results of a keyword search are (also) displayed in the graph view of the GUI. In this graph view the author can browse ontological relationships.
- Visualising common properties between resources (RQ1.4) is created by grouping resources that have common properties into clusters. The clusters are browse-able in the GUI.

The main approach to making searching easier for the author is creating a GUI that gives the author overview in the relationships of data elements.

The designs for this research question were further specified in:

- Paragraph 3.5 Searching for resources
- Paragraph 3.10 RMT Graphical User Interface
- Paragraph 4.3 RMT application

[22] The project goals are directly derived from the research questions and therefore the evaluation of the goals is similar to that of the research questions.

Research question 2: How to manage resources efficiently?

- 2.1 What management tasks to perform on resources?
- 2.2 How to use the ontology (domain/course/resource structure) to aid in managing resources.
- 2.3 How to position the resources in the overall OntoAIMS ontology; relating resources to other objects (resources/concepts/courses).
- 2.4 How to suggest helpful information to the user.

Evaluation of research question 2:
To efficiently manage resources (RQ2) these parts of the RMT were designed:

- The management tasks on resources (RQ2.1) that the system needs to provide to the author are create, edit, delete and view as specified in the RMT Use Case.
- The use of the ontology in managing resources (RQ2.2) is not supported in this project. The focus of the project was on using the ontology in searching. Using the ontology in editing can be worked out in future work.
- The placement of resources in the OntoAIMS ontology (RQ2.3) is defined by the open data structure of OntoAIMS. Each resource in the resource pool is referable from each course and each domain concept. This supports maximum reuse of resource across courses.
- Suggesting information to the author (user) (RQ2.4) is done by:
  - Showing relations in the GUI by 'highlighting' items on 'mouse-over-events'.
  - Showing example data and hints where the author has to input data.
  - Presenting the author with uniform formatted warnings and errors

The designs for this research question were further specified in:

- Paragraph 3.2 Use Case
- Paragraph 3.6 Architecture
- Paragraph 4.3 RMT application

Research question 3: How to model resources in OntoAIMS?

- 3.1 What to use of which open data standards.
- 3.2 How to use the semantics of the data.

Evaluation of research question 3:
To model resources (RQ3) these parts of the RMT were designed:

- For the metadata specification for OntoAIMS resources (RQ3.1) LOM was chosen from several open web standards for metadata. The specification for the RMT had to fit in with the specifications chosen for the DMT and the CST.
- Using the semantics of the metadata (RQ3.2) is not supported in this project. The focus of the project was on using the ontology in searching. Semantically gathering metadata to assist the author can be worked out in future work.

The designs for this research question were further specified in:

- Paragraph 3.3 Resources
- Paragraph 4.3 RMT application

## *5.4 Timing the project*

This graduation project ran much longer than the usual period of one year. Many factors contributed to this delay. These (delay factors) can be grouped into computer related, organizational, personal factors and difficult project parts:

Computer related:
Several weeks were spent on:

- Installing and re-installing software. First the software was installed on my laptop, then on a faster laptop, later in a new profile on the same laptop and finally on a pc. This because of old hardware and changes in the network setup.
- Fixing problems with viruses and updating licences.
- Having the helpdesk fix broken hardware.

Organizational:
The project suffered several delays on:

- Miscommunication between the project members.
- Disagreement on several designs between the project members.

Difficult project parts
Some parts took very long because:

- This was my first large project programming in a Java environment. Previous projects in C, Visual Basic, several scripting languages and some Delphi helped in understanding the basics but some training exercises were necessary.
- I spent a long time searching for a standard syntax to describe query transformation. Searching on the Internet, browsing through some Information Retrieval books and asking colleagues did not give a useful result.
- Inserting more than one hundred resources and several concept links into the RMT took a long time. This to thoroughly test searching and editing functions.
- Inexperience with UML and design modelling.
- It took more then a few redesigns and builds to create a working scaleable user interface in Java.
- The thesis took several rewrites to cover all the information on the project.

Personal:
Several weeks got lost on being sick and relatives dieing.
Several weeks were used for taking courses, studying for and taking exams and for taking a few days off.

# 6. Recommendations

While the RMT is already quite elaborate, there still is a lot that can be done to extend the application. The time for a project is almost always limited, so not every option that would serve the author could be made. The recommendations for future work are divided into three categories: Visualisation, Data entry and Data conversion.

## *6.1 Visualisation*

Overcrowding of the graph

The main visualization aspect of the RMT is the relationship graph. In this graph the relation between resources, clusters and concepts is visualized. To create overview different objects in the graph have different colors: a resource is green, a concept is blue and a cluster is orange. Despite the color coding and that the graph shows only objects that are 'one relation away' the graph might get crowded.

Recommendation(s):

To reduce the crowding of the graph view one or more of these options can be built:

- Letting the author choose a filtered view.
  Reducing the objects in the graph by showing only part of the relations. The author selects a view filter to display only connected concepts or a view filter that omits sub-clusters, etc.
- Dividing the graph in more rings.
  The current graph view shows all connected objects in an oval shape around the selected object. Showing the connected objects in two or three rings around the selected object could create more overview.

## *6.2 Data entry*

Common words for query filtering

To filter 'common words'[23] from the author's search query a list of common words is used. This list of common words is embedded in the RMT.

Recommendation(s):

For more control over the filtering of search queries the author needs to be able to manage this list of common words. To manage the common word list a new GUI element in the RMT user interface is needed. This interface can be similar to the currently available functionality for adding relationship types.

[23] See *paragraph 3.5.2 Filtering the user query* for a more elaborate definition of common words in searches.

Query spelling correction

When entering a query an author might not know the correct spelling of a search word or the author might make a spelling mistake.

Recommendation(s):

To counteract bad search results, because of misspelling, WordNet and / or the OntoAIMS ontology can be used to deduce words that give better (most likely more) search results. Both Google [Google] and the Internet Movie Data Base [IMDB] use spelling 'correction' to aid the user in searching. See Figure 42 and Figure 43 for examples. In both examples the user query is executed on the system and results are shown. If words that look like words in the query yield more results then those words with more results are suggested to the user.

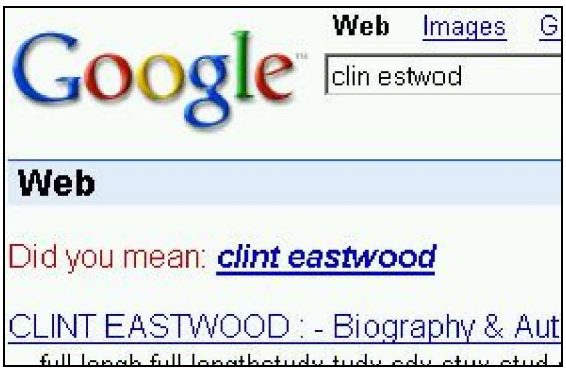


**Figure 42, Google spelling correction**

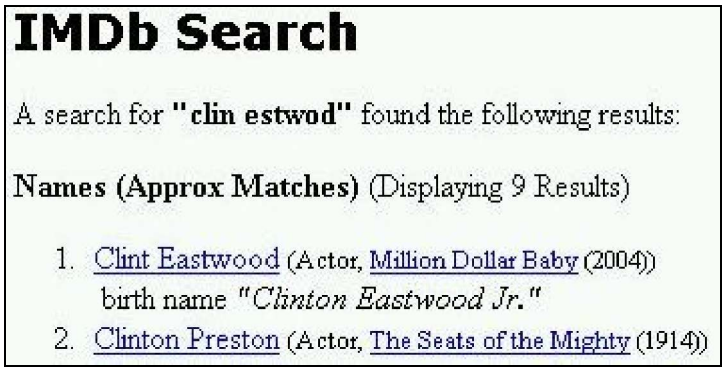


**Figure 43, IMDB spelling correction**

Gathering metadata by URI
The creation of new resources (metadata sets) is still much work, the author has to type in more then a few descriptions and select some from lists. The more metadata information an author enters into the system for a resource the more useful that resource becomes.

Recommendation(s):
A subsystem could be build where the user only gives the URI of the resource to add and the system would analyse the URI to try to fill in default information so the author does not have to. If the URI is of a website some data can be retrieved automatically like the title or the author[24].

Next to querying the URI some guesses for metadata can be made on evaluating the type of the URI. If the URI starts with "http:" the learning type is probably 'hypertext document' and if the URI starts with 'ISBN:' the learning type is most likely 'book'.

## 6.3 Data conversion

The resource pool is formatted using the LOM and DC specification. The OntoAIMS resource pool can be copied on another OntoAIMS application to enable that application to use the same resources. At the moment there is no facility to merge OntoAIMS resource pools or integrate other metadata sets into the resource pool.

Recommendation(s):
For external use of the resource data an export facility is useful to create a representation of the resource pool in another open metadata specification or simply make a copy of the resource pool (backup or use elsewhere).

To enable the addition from resources from other OntoAIMS resource pools a merge facility can be built.

To enable the addition from resources from other metadata specifications an import filter can be built.

[24] The author, description, keywords, etc of a webpage are often written in the page in 'meta'-tags. Example: for www.tue.nl this tag is present: <meta name="copyright" content="(c) TU/e 2005" />.

# References


| | |
|---|---|
| **[AIMS]** | **Title:** AIMS: Adaptive Information System for Management of Learning Content |
| **Author:** | AIMS project homepage |
| **URI:** | http://wwwis.win.tue.nl:8082/aims/<br>http://www.win.tue.nl/~laroyo/AIMS/ |
| **[Aroyo]** | **Title:** Task-oriented approach to information handling support within web-based education |
| **Author:** | Lora Aroyo |
| **URI:** | ISBN:9036516900 |
| **Edition:** | 2001, PrintPartners Ipskamp, Enschede |
| **[Aroyo2]** | **Title:** - |
| **Author:** | Dr. L.M. Aroyo |
| **URI:** | http://www.win.tue.nl/~laroyo/ |
| **[Aroyo+ Mizoguchi]** | **Title:** Process-aware Authoring of Web-based Educational Systems |
| **Author:** | Lora Aroyo & Riichiro Mizoguchi |
| **URI:** | http://www.sw-wl03.bessag.net/images/Paper%20PDF/Presentation%20P1.pdf |
| **[AquaBrowser]** | **Title:** AquaBrowser Library |
| **Author:** | Medialab Solutions BV |
| **URI:** | http://www.medialab.nl/ |
| **[Bastiaans]** | **Title:** OntoAIMS: Ontological Approach to Courseware Authoring |
| **Author:** | Siemen Bastiaans |
| **URI:** | - |
| **[BorlandXML]** | **Title:** The XML Developer's Guide |
| **Author:** | Borland |
| **URI:** | http://info.borland.com/techpubs/jbuilder/jbuilder8/xml/xmlintro.html |
| **[CanCore]** | **Title:** CanCore |
| **Author:** | The CanCore Initiative |
| **URI:** | http://www.cancore.ca/en/ |
| **[ClusterMap]** | **Title:** Cluster Map Library 2005.1 |
| **Author:** | Aduna |
| **URI:** | http://aduna.biz/products/technology/clustermap/index.html |
| **[CSDGM]** | **Title:** Content Standard for Digital Geospatial Metadata (CSDGM) |
| **Author:** | The Federal Geographic Data Committee |
| **URI:** | http://www.fgdc.gov/metadata/contstan.html |
| **[DAML]** | **Title:** The DARPA Agent Markup Language Homepage |
| **Author:** | The Defense Advanced Research Projects Agency (DARPA) |
| **URI:** | http://www.daml.org/ |
| **[DAMLOIL]** | **Title:** DAML+OIL (March 2001) |
| **Author:** | The Defense Advanced Research Projects Agency (DARPA) |
| **URI:** | http://www.daml.org/2001/03/daml+oil-index.html |

| | |
|---|---|
| **[DC]** | **Title:** Dublin Core Metadata Element Set, Version 1.1: Reference Description |
| **Author:** | Dublin Core Metadata Initiative |
| **URI:** | http://www.dublincore.org/documents/dces/ |

| | |
|---|---|
| **[GEM]** | **Title:** The Gateway to Educational Materials |
| **Author:** | U.S. Department of Education's National Library of Education |
| **URI:** | http://www.thegateway.org/ |

| | |
|---|---|
| **[gbiv]** | **Title:** Uniform Resource Identifier (URI): Generic Syntax |
| **Author:** | T. Berners-Lee |
| **URI:** | http://gbiv.com/protocols/uri/rev-2002/rfc2396bis.html |

| | |
|---|---|
| **[Google]** | **Title:** Google |
| **Author:** | Google |
| **URI:** | www.google.com |

| | |
|---|---|
| **[Hubens]** | **Title:** Authoring support for domain modelling in concept-based educational system |
| **Author:** | J.L.P. Hubens |
| **URI:** | - |

| | |
|---|---|
| **[IMDB]** | **Title:** The Internet Movie Database, Earth's Biggest Movie Database |
| **Author:** | Internet Movie Database Inc. |
| **URI:** | http://www.imdb.com/ |

| | |
|---|---|
| **[IMS]** | **Title:** IMS Meta-Data (IMS) |
| **Author:** | IMS Global Learning Consortium |
| **URI:** | http://www.imsglobal.org/ |

| | |
|---|---|
| **[Jena]** | **Title:** Jena – A Semantic Web Framework for Java |
| **Author:** | Hewlett-Packard Development Company, LP |
| **URI:** | http://jena.sourceforge.net/ |

| | |
|---|---|
| **[JGraph]** | **Title:** The Java Graph Visualization Library |
| **Author:** | JGraph, Ltd. |
| **URI:** | http://www.jgraph.com/ |

| | |
|---|---|
| **[JWNL]** | **Title:** Java WordNet Library |
| **Author:** | John Didion |
| **URI:** | http://jwordnet.sourceforge.net/<br>http://sourceforge.net/projects/jwordnet |

| | |
|---|---|
| **[JWordNet]** | **Title:** JWordNet |
| **Author:** | Oliver Steele |
| **URI:** | http://jwn.sourceforge.net/ |

| | |
|---|---|
| **[LOM]** | **Title:** Final 1484.12.1 LOM Draft Standard Document |
| **Author:** | IEEE Learning Technology Standards Committee |
| **URI:** | http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf |

| | |
|---|---|
| **[MARC]** | **Title:** MARC Standards |
| **Author:** | The Library of Congress - Network Development and MARC standards office |
| **URI:** | http://www.loc.gov/marc/ |

| | |
|---|---|
| **[MODS]** | **Title:** Metadata Object Description Schema (MODS) |
| **Author:** | The Library of Congress' Network Development and MARC Standards Office |
| **URI:** | http://www.loc.gov/standards/mods/ |
| **[OWL]** | **Title:** OWL Web Ontology Language Reference |
| **Author:** | W3C |
| **URI:** | http://www.w3.org/TR/owl-ref/ |
| **[Query]** | **Title:** Query modification and. relevance feedback |
| **Author:** | Ian Ruthven |
| **URI:** | http://www.cis.strath.ac.uk/~fabioc/04-mia/lects/05.pdf |
| **[RDF]** | **Title:** Resource Description Framework (RDF) |
| **Author:** | The World Wide Web Consortium (W3C) |
| **URI:** | http://www.w3.org/RDF/ |
| **[RDFS]** | **Title:** RDF Vocabulary Description Language 1.0: RDF Schema |
| **Author:** | The World Wide Web Consortium (W3C) |
| **URI:** | http://www.w3.org/TR/rdf-schema/ |
| **[RFC2048]** | **Title:** Multipurpose Internet Mail Extensions (MIME) |
| **Author:** | Network Working Group |
| **URI:** | http://rfc.net/rfc2048.html |
| **[Saxon]** | **Title:** SAXON The XSLT and XQuery Processor |
| **Author:** | Michael Kay |
| **URI:** | http://saxon.sourceforge.net/ |
| **[RIO]** | **Title:** Cisco Systems Reusable Information Object Strategy |
| **Author:** | Cisco systems |
| **URI:** | http://www.cisco.com/warp/public/779/ibs/solutions/learning/whitepapers/el_cisco_rio.pdf |
| **[UML]** | **Title:** Agile Models Distilled: Potential Artefacts for Agile Modelling (Overview of UML models) |
| **Author:** | Scott W. Ambler |
| **URI:** | http://www.agilemodeling.com/artifacts/ |
| **[WNJN]** | **Title:** WordNet JNI Java Native Support |
| **Author:** | Bernard BOU |
| **URI:** | http://wnjn.sourceforge.net/ |
| **[WordNet]** | **Title:** WordNet, a lexical database for the English language |
| **Author:** | Cognitive Science Laboratory at Princeton University |
| **URI:** | http://www.cogsci.princeton.edu/~wn/ |
| **[Xerces]** | **Title:** The Apache Xerces2 parser |
| **Author:** | The Apache Software Foundation |
| **URI:** | http://xml.apache.org/xerces2-j/index.html |

# Appendices

Appendix figures:

## *Appendix 1: RMT metadata tags*

For the notation of OntoAIMS resources a subset of data definition labels (tags) was chosen from the Dublin Core (DC) and Learning Objects Metadata (LOM) standards. That subset of tags with descriptions and examples is shown here.

For some overview the metadata is grouped into 5 sets, these set names are also used in the RMT GUI to divide the input fields over the tabs of the metadata edit / view dialog. The groups are: names, description, format, requirement & education.

"NS" = name space[25],
"Tag" = metadata field name from name space.
Metadata set OntoAIMS resource:

Group Names:

| NS: Tag: | - | Description: | Resource identifier |
|---|---|---|---|
| Name: | URI | Example: | "http://wwwis.win.tue.nl/AIMS" |

| NS: Tag: | Dc: Title | Description: | Title for the resource |
|---|---|---|---|
| Name: | Title / name | Example: | "List of university courses" |

| NS: Tag: | Dc: Source | Description: | Where the resource was found / linked from |
|---|---|---|---|
| Name: | Source | Example: | "http://company.org/personal_homepage.htm" |

| NS: Tag: | Dc: Author | Description: | The person/company/… that made the resource |
|---|---|---|---|
| Name: | Author | Example: | "Sir Arthur Conan Doyle" |

| NS: Tag: | Dc: Date | Description: | Date on which the resource was created by the author |
|---|---|---|---|
| Name: | Creation date | Example: | "may 2000" or<br>"27-03-1922" |

| NS: Tag: | Rmt: ResourceDate | Description: | Date (time) when the resource metadata was entered in the resource pool |
|---|---|---|---|
| Name: | Metadata date | Example: | "2004.11.12 14:02" |

| NS: Tag: | Rmt: LastResDate | Description: | Date (time) when the resource metadata was last saved to the resource pool (last changed) |
|---|---|---|---|
| Name: | **Last metadata date** | Example: | "2004.12.11 09:14" |

| NS: Tag: | Dc: Publisher | Description: | The name of the metadata editor who entered the resource metadata into the resource pool |
|---|---|---|---|
| Name: | OntoAIMS editor | Example: | "Default editor" |

| NS: Tag: | Rmt: LastPublisher | Description: | The name of the metadata editor who last changed the resource metadata |
|---|---|---|---|
| Name: | **Last OntoAIMS editor** | Example: | "Jurgen" |

[25] The namespace the short name of the metadata specification where the metadata tag was defined. DC = Dublin Core, LOM = Learning Objects Metadata and RMT = OntoAIMS Resource Management Tool.

Group Description:

| NS: Tag: | Dc: Description | Description: | A description of the contents of the resource |
|---|---|---|---|
| Name: | Description | Example: | "Painting of a ship on sea carrying troops to war" |

| NS: Tag: | Dc: Coverage | Description: | A description of the setting for the resource, a comment on the time setting, cultural information, geographical context, region, etc. |
|---|---|---|---|
| Name: | **Coverage** | Example: | "Renaissance painting from early France" |

| NS: Tag: | Rmt: Keyword (Lom: Keyword, Rmt: Dmt-link ) | Description: | Combination of keyword for the resource and URI of a concept in a domain from the domain editor that represents the keyword |
|---|---|---|---|
| Name: | **Keyword (set)** | Example: | ("Boat", "http://wwwis.win.tue.nl/domain/Vehicles#Boat") |

| NS: Tag: | Dc: Language | Description: | The language in which the resource is written |
|---|---|---|---|
| Name: | **Language** | Example: | "En" for English |

Group Format:

| NS: Tag: | Lom: Format | Description: | Technical type of resource object[26] |
|---|---|---|---|
| Name: | Format | Example: | "Text/plain" or "Application/excel" or "Image/picture" |

| NS: Tag: | Lom: Size | Description: | File size of the resource in bytes |
|---|---|---|---|
| Name: | Byte size | Example: | "20.154" or "1024" |

| NS: Tag: | Rmt: LogicalSize | Description: | Volume / size / range of resource in words |
|---|---|---|---|
| Name: | Logical size | Example: | "Couple of pages" or "One hour of reading" |

| NS: Tag: | Lom: Metadatascheme | Description: | Type / version of metadata definitions |
|---|---|---|---|
| Name: | **Metadata scheme** | Example: | "LOMv1.0" |

Group Requirement:

| NS: Tag: | Lom: Requirement (Lom: Type, Lom: Name, Lom: Minversion ) | Description: | Technical resource requirement as set of:<br>1. Type of requirement<br>2. Name of requirement type<br>3. Minimal required version of name |
|---|---|---|---|
| Name: | Requirement (set) | Example: | ("Browser", "Internet explorer", "5") |

| NS: Tag: | Rmt: Relation (Lom: Kind, Lom: Resource, Lom: Description ) | Description: | Relation to another resource as set of:<br>1. Kind of relation<br>2. URI of related resource<br>3. Description of relation |
|---|---|---|---|
| Name: | Relation (set) | Example: | ("part-of", "ISBN:00012545", "chapter of this book") |

[26] Format is specified in MIME types based on IANA Registration (see [RFC2048]) or non-digital. As described in [LOM]

Group Education:

| NS: Tag: | Lom: Interact_type | Description: | Mode of learning supported by the resource |
|---|---|---|---|
| Name: | Interaction type | Example: | "Active" |
| | | Options: | Names of modes are defined in vocabularies below, #1 |

| NS: Tag: | Lom: Learningres_type | Description: | Type of learning that the resource offers |
|---|---|---|---|
| Name: | **Learning resource type** | Example: | "Exercise" or "Diagram" |
| | | Options: | Types are defined in vocabularies below, #2 |

| NS: Tag: | Lom: Interact_level | Description: | Degree of interactivity of resource with the user / reader / etc. |
|---|---|---|---|
| Name: | Interaction level | Example: | "Medium" |
| | | Options: | Levels are defined in vocabularies below, #3 |

| NS: Tag: | Lom: Sem-density | Description: | Degree of information content (effect) in resource |
|---|---|---|---|
| Name: | Semantic density | Example: | "High" |
| | | Options: | Densities are defined in vocabularies below, #3 |

| NS: Tag: | Lom: Context | Description: | Target learning environment or expected education level of user for this resource |
|---|---|---|---|
| Name: | Context | Example: | "Higher education" or "Training" |
| | | Options: | Context types are defined in vocabularies below, #4 |

| NS: Tag: | Lom: Difficulty | Description: | Estimate of learning difficulty for target audience (context) |
|---|---|---|---|
| Name: | Difficulty | Example: | "Easy" |
| | | Options: | Difficulty stages are defined in vocabularies below, #5 |

Notes:
All metadata fields occur once per resource except for Lom: requirement, Rmt: relation and Rmt: keyword. A resource can have zero or more keywords (keyword sets), relations (relation sets) and / or requirements (requirement sets).

The fields URI, title, description, author and date are required. Other fields are optional but the more information the metadata contains the better it can be searched / managed / used.

OntoAIMS metadata vocabularies:

| # | Vocabulary | Members |
|---|---|---|
| 1 | Interaction | Unknown, Active, Expositive, Mixed |
| 2 | Resource type | Undefined, Assignment, Audio clip, Book, Case, Chapter, Description, Diagram, Example, Exam, Exercise, Figure, Graph, Hypertext document, Hypermedia document, Index, Image/Photo, Journal, Lecture, Manual, Narrative text, Paper, Problem statement, Procedure, Self assessment, Simulation, Slide, Table, Questionnaire, Reader, Test, Video clip |
| 3 | Height | Unknown, Very low, Low, Medium, High, Very high |
| 4 | Training | Undefined, Primary education, Secondary education, Higher education, University, Technical school, Professional, Continuous, Vocational |
| 5 | Toughness | Unknown, Very easy, Easy, Medium, Difficult, Very difficult |

## *Appendix 2: RMT resource example*

The following frame contains an example (without any real data but with the full name of the element in bold) of an OntoAIMS resource with the name space definitions, one keyword, one relation and one requirement.

```
<rdf:RDF
    xmlns="http://purl.org/rss/1.0/"
    xmlns:rmt="http://localhost/ontoaims/rmt/schemas/1.0/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:lom="http://kmr.nada.kth.se/el/ims/schemas/lom-general/" >

  <rdf:Description rdf:about="resource_uri">
    <dc:Title>Title / name</dc:Title>
    <dc:Source>Source URI or description</dc:Source>
    <dc:Author>Author / company name</dc:Author>
    <dc:Date>Resource date</dc:Date>
    <rmt:Resourcedate>resourcepool date</rmt:Resourcedate>
    <rmt:Lastresdate>resourcepool date</rmt:Lastresdate>
    <dc:Publisher>AIMS editor</dc:Publisher>
    <rmt:Lastpublisher>AIMS editor</rmt:Lastpublisher>
    <dc:Description>Description of resource contents</dc:Description>
    <dc:Coverage>Description of time/place for resource</dc:Coverage>
    <rmt:Keyword rdf:resource="Keyword_uri_1"/>
    <rmt:Keyword rdf:resource="Keyword_uri_2"/>
    <rmt:Keyword rdf:resource="Keyword_uri_3"/>
    <dc:Language>short language identifier</dc:Language>
    <lom:Format>Format / type</lom:Format>
    <lom:Size>Size in bytes</lom:Size>
    <rmt:LogicalSize>Size in terms</rmt:LogicalSize>
    <lom:Metadatascheme>LOM version string</lom:Metadatascheme>
    <lom:Requirement rdf:resource="Requirement_uri_1"/>
    <rmt:Relation rdf:resource="Relation_uri_1"/>
    <lom:Interact_type>Mode of learning</lom:Interact_type>
    <lom:Learningres_type>Type of learning object</lom:Learningres_type>
    <lom:Interact_level>User interaction level</lom:Interact_level>
    <lom:Sem_density>Scale of compactness of resource</lom:Sem_density>
    <lom:Context>Target education</lom:Context>
    <lom:Difficulty>Scale of difficulty</lom:Difficulty>
  </rdf:Description>

  <rdf:Description rdf:about="Keyword_uri_1">
    <rmt:Dmt-link>URI of dmm concept</rmt:Dmt-link>
    <lom:Keyword>Keyword name</lom:Keyword>
  </rdf:Description>

  <rdf:Description rdf:about="Requirement_uri_1">
    <lom:Type>Requirement type</lom:Type>
    <lom:Name>Name of requirement of type</lom:Name>
    <lom:Minversion>Number or specification or id of name</lom:Minversion>
  </rdf:Description>

  <rdf:Description rdf:about="Relation_uri_1">
    <lom:Kind>Type of relation</lom:Kind>
    <lom:Resource>Resource URI </lom:Resource>
    <lom:Description>Description of relation</lom:Description>
  </rdf:Description>

</rdf:RDF>
```

## *Appendix 3: Conceptual model*

The conceptual model defines the data structure of the object(s) that the system contains / handles. This conceptual model defines the OntoAIMS resource and its metadata components.

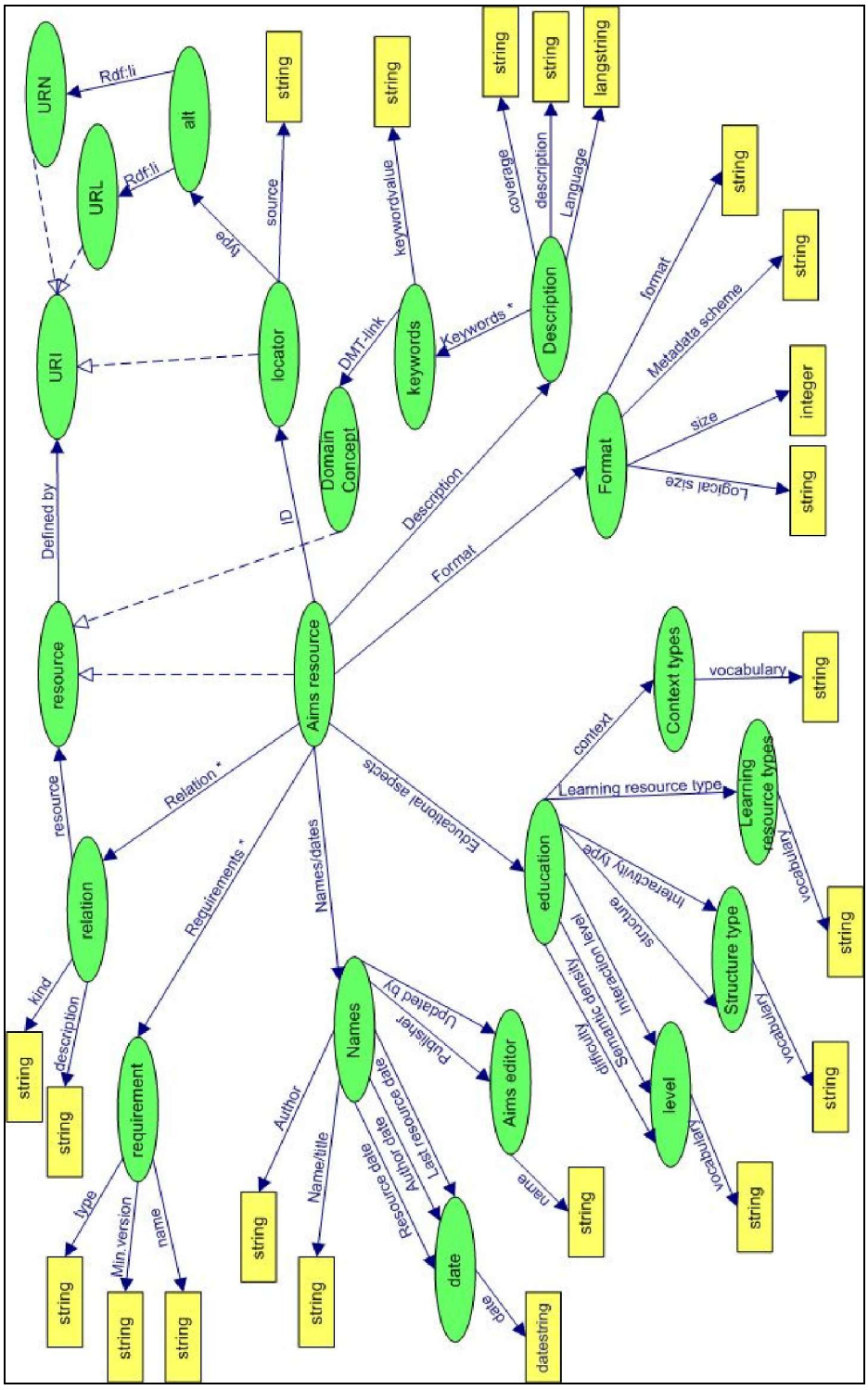


**Figure A 1, Conceptual model**

## *Appendix 4: URI specification*

A Uniform Resource Identifier (URI) provides a simple and extensible means for identifying a resource. This specification of URI syntax and semantics is derived from concepts introduced by the World Wide Web global information initiative, whose use of such identifiers dates from 1990 and is described in "Universal Resource Identifiers in WWW" (RFC 1630)[27].

A common misunderstanding of URI's is that they are only used to refer to accessible resources. In fact, the URI alone only provides identification; access to the resource is neither guaranteed nor implied by the presence of a URI.

A URI can be 1 of 2 types:
"Uniform Resource Locator" (URL) refers to the subset of URI's that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its "network location").
"Uniform Resource Name" (URN) refers to any other URI with the properties of a name.

A URI has a fixed structure composed of at least a scheme name (or an abbreviation of a scheme name) and a path, as shown in the next figure:

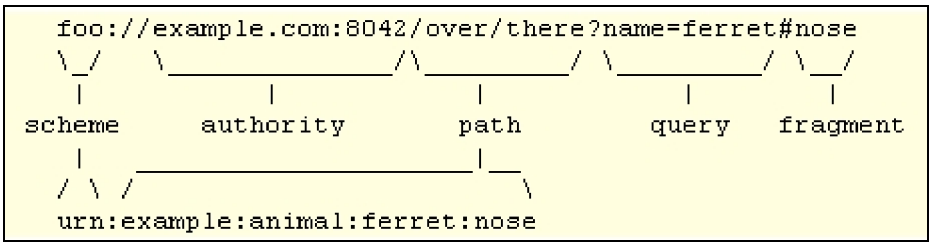


**Figure A 2, URI schema**

Image source: http://gbiv.com/protocols/uri/rev-2002/rfc2396bis.html
Author: T. Berners-Lee

URI examples:
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2

[27] http://gbiv.com/protocols/uri/rev-2002/rfc2396bis.html#RFC1630

## *Appendix 5: Class diagrams*

The architecture design of the RMT translates to the following UML classes. Class diagrams specify the architectural components of the application, the most important functions and data for each class and the relations between classes.

The OntoAIMS authoring environment consists of four modules:

- Domain Modeling Tool
- Course Sequencing Tool
- Resource Management Tool
- Dictionary

These modules are separate but interact using each other's interface. The 'ModuleHandler' provides the access to the interfaces of the authoring modules.

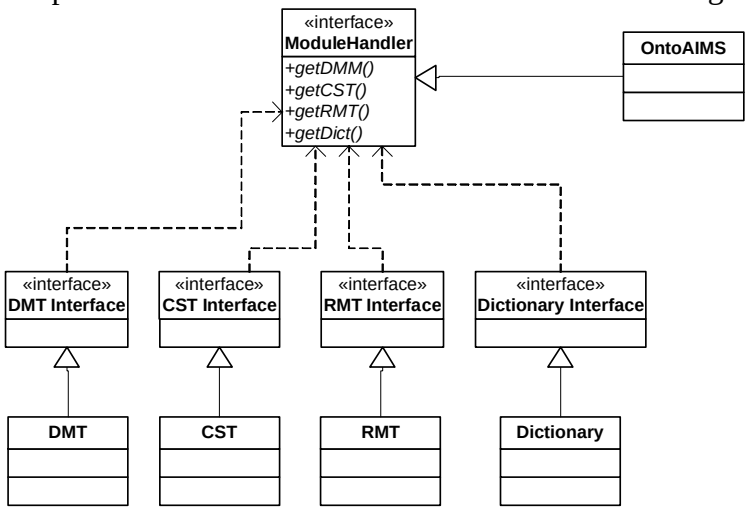


**Figure A 3, OntoAIMS main class structure**

Within this project the RMT module and the Dictionary module are designed and created.

The Dictionary structure provides a 'wrapper' class on the JWNL WordNet library to access specific dictionary properties from WordNet.

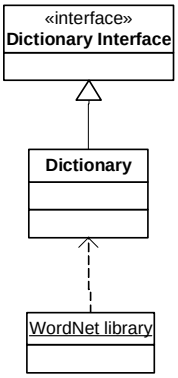


**Figure A 4, Dicionary class structure**

The RMT class structure represents the RMT architecture with the RMT interface, 3 components (P&F, Search and Resources) and the user interface (GUI component). Added is the RMT (main) class to link the interface to the components and the Types class for the specifications of resource types.

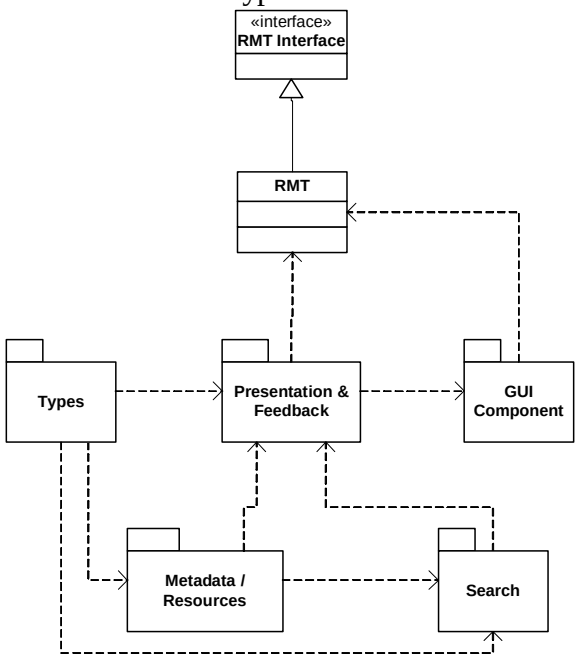


**Figure A 5, RMT main class structure**

The GUI component class encapsulates the components that make up the user interface(s).

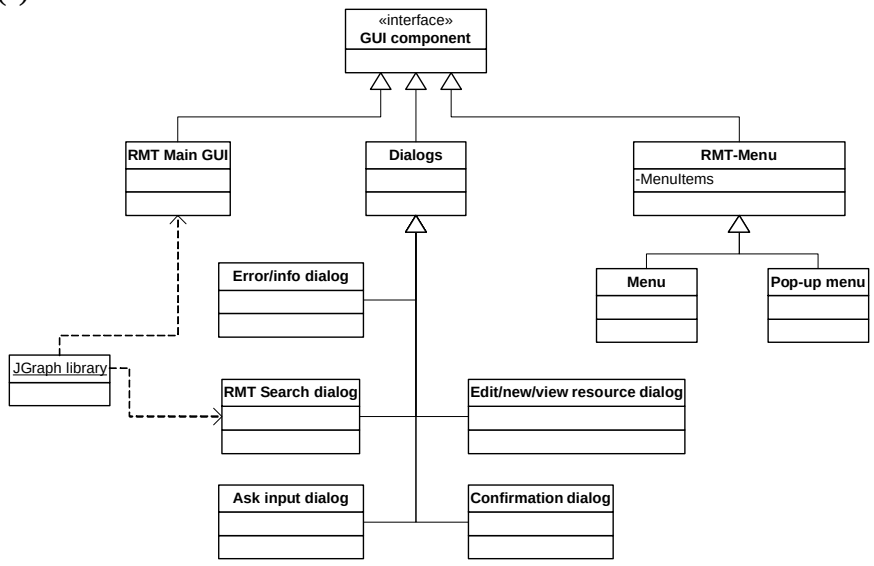


**Figure A 6, RMT GUI class structure**

The resources / metadata class is split into an interface splitting of the functions accessible by other components and the internal metadata functions

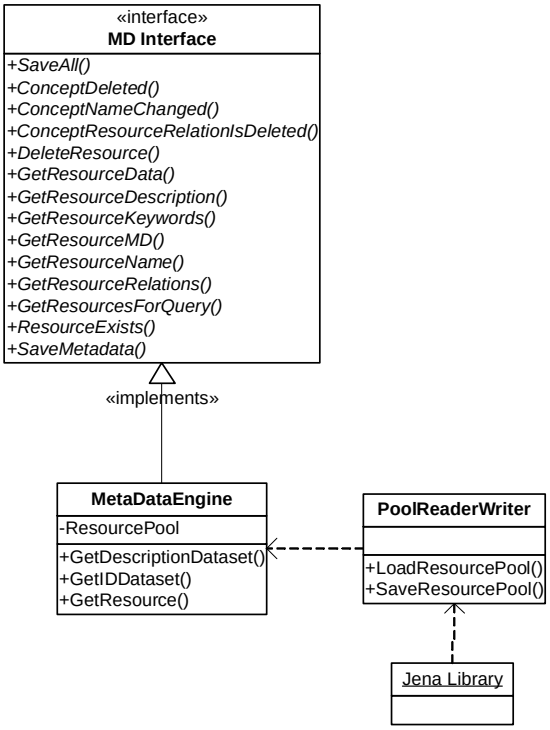


**Figure A 7, Resources / metadata class structure**

The same goes for the search class.

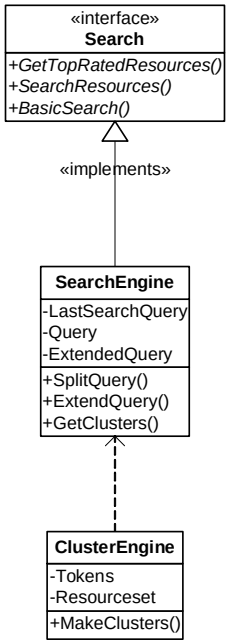


**Figure A 8, Search class structure**

The presentation & feedback class structure. Split up into Dispatch, Presentation and Feedback as described in the architecture.

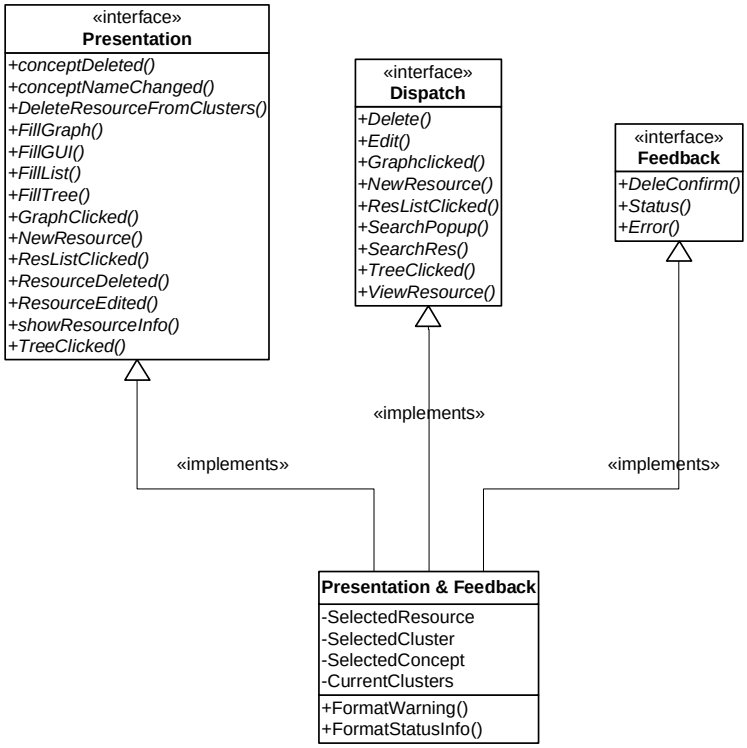


**Figure A 9, Presentation & Feedback class structure**

The types classes define the types used in RMT to exchange data in the system.

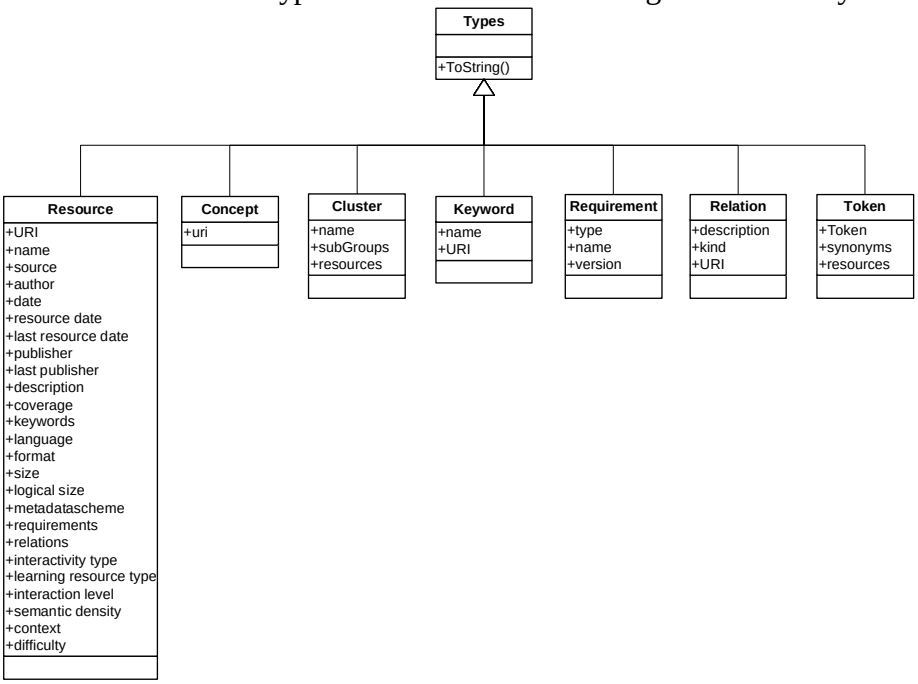


**Figure A 10, Types class structure**

# Appendix 6: JavaDOC specification

The documentation for the application is entered with the code in Java. Using the JavaDOC coding tool – developed by Sun - special comment sections are inserted in the code. The JavaDOC tool parses the code and generates a structured HTML file for each Java class describing the commented sections, function headers and Java package order.

Example code:

```
/**
 * <p>Title: Dispatch interface</p>
 * <p>Description: part of Presentation & Feedback</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: <a href="http://www.tue.nl">Eindhoven University of Technology</a></p>
 * @author <a href="mailto:j.t.j.rijsdijk@tue.nl">Jurgen Rijsdijk</a>
 * @version 1.0
 */

public interface DispInterface {

  /**
   * dispatch delete resource command
   * @param ResourceId resource to delete
   */
  public void Delete(String ResourceId);

  /**
   * show edit dialog
   * @param ResourceId resource to edit
   */
  public void Edit(String ResourceId);
```

Generated JavaDOC HTML fragment:

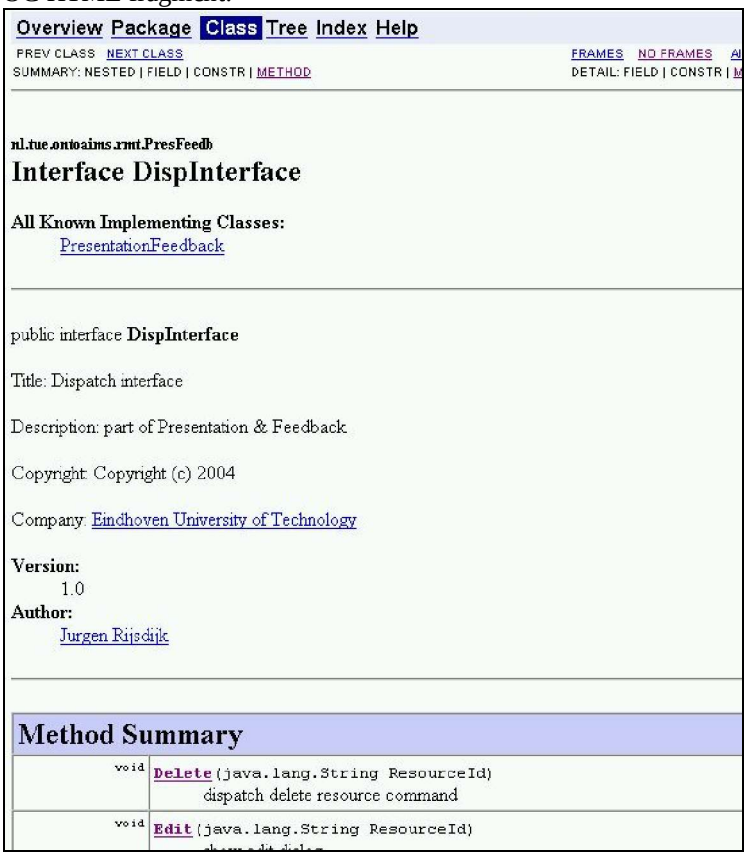


**Figure A 11, JavaDOC example**

# *Appendix 7: Module communication*

Each of the three OntoAIMS modules has a communication interface with which it gives access to the public functionality for the other modules.

## Appendix 7.1 RMT module communication

R= request. M= Message.

Requests & messages from RMT to CST

| R/M | *Description* | **Parameters** | **Response** |
|---|---|---|---|
| M | A resource is deleted | Resource URI | - |
| M | The identifier of a resource changed | Old URI, new URI | - |

Requests & messages from CST to RMT

| R/M | *Description* | **Parameters** | **Response** |
|---|---|---|---|
| R | Get the description for a resource | Resource URI | Description text |
| R | Get the keywords for a resource | Resource URI | List of keywords |
| R | Get the name for a resource | Resource URI | Name text |
| R | Does a resource exist | Resource URI | Yes or no (true/false) |

Requests & messages from RMT to DMM

| R/M | *Description* | **Parameters** | **Response** |
|---|---|---|---|
| M | A resource is deleted | Resource URI | - |
| M | The identifier of a resource changed | Old resource URI, New resource URI | - |
| M | Relation between concept and resource was removed in the resource metadata | Resource URI, Concept URI | - |
| R | Does a concept exist | Concept URI | Yes or no (true/false) |
| R | Get resources connected to a concept | Concept URI | List of resource URI |
| R | Get concepts connected to a concept | Concept URI | List of concept URI |
| R | Get concepts connected to a resource | Resource URI | List of concept URI |
| R | Is a resource used in a domain | Resource URI | Yes or no (true/false) |
| R | Is there a concept for a name | Name text | List of concept URI |
| R | Search for a concept in a dialog | - | URI for concept selected in dialog |

Requests & messages from DMM to RMT

| R/M | *Description* | **Parameters** | **Response** |
|---|---|---|---|
| R | Get the description for a resource | Resource URI | Description text |
| R | Get the keywords for a resource | Resource URI | List of keywords |
| R | Get the name for a resource | Resource URI | Name text |
| R | Does a resource exist | Resource URI | Yes or no (true/false) |
| R | Search for a resource in a dialog | - | URI of resource selected in dialog |
| R | Get a list of a maximum length of resources matching a word | Search word, Number | A list of at most 'Number' items of resource URI where the referenced resources were found using the 'Search word'. |

Requests & messages from RMT to Dictionary

| R/M | *Description* | **Parameters** | **Response** |
|---|---|---|---|
| R | Get the synonyms for a word | Word | List of synonyms |

Requests & messages from Dictionary to RMT

For the moment there are no known messages or requests for the Dictionary to send to the RMT.

## Appendix 7.2 Communication interfaces

The communication interfaces are displayed in the following four chapters in JavaDOC format.

### 7.2.1 CST communication interface

Public interface **CSTInterface**

*Interface to make the Course Sequencing Module (CSM) accessible to other OntoAims Modules. It provides methods for accessing course information, and methods to keep the courses consistent with the information in the other modules.*

**Version:**

1.3

**Author:**

Johan Hubens,, Jurgen Rijsdijk,
Eindhoven University of Technology.

---

| Method Summary | |
|---|---|
| void | **ConceptDeleted**(java.lang.String ConceptID)<br>Method called by the Domain Modeling Module (DMM) when a concept is deleted in the DMM. |
| java.lang.String[] | **ConceptInCourses**(java.lang.String ConceptId)<br>list the courses where this concept is used |
| void | **ConceptNameChanged**(java.lang.String oldName, java.lang.String newName)<br>message that the name of a concept had changed |
| boolean | **ConceptUsed**(java.lang.String ConceptID)<br>Answer whether a concept from a domain in the domain modeling module (DMM), identified by the String ConceptID is used in a course. |
| boolean | **dataChanged**()<br>answer if the course data changed since last save/open |
| void | **ResourceDeleted**(java.lang.String ResourceID)<br>Method called by the Resource Management Module (RMM) when a resource is deleted from the resource pool. |
| void | **ResourceIDChanged**(java.lang.String oldID, java.lang.String newID)<br>message that the id (uri) of a resource has changed |
| java.lang.String[] | **ResourceInCourses**(java.lang.String ResourceId)<br>list of courses where resource is used |
| boolean | **ResourceUsed**(java.lang.String ResourceID)<br>Answer whether a resource from the resource pool, identified by the String ResourceID is used in a course. |
| void | **saveAll**()<br>request from main program to save all course data to disc. |

### 7.2.2 DMM communication interface

Public interface **DMMInterface**

Interface to make the Domain Modeling Module (DMM) accessible to other OntoAims Modules. It provides methods for accessing domain information, and methods to keep the domain models consistent with the information in the other modules.

**Version:**

1.0

**Author:**

Johan Hubens,
Eindhoven University of Technology.

| Method Summary | |
|---|---|
| boolean | **ConceptExists**(java.lang.String ConceptID)<br>Answer whether a concept identified by the String ConceptID exists. |
| boolean | **dataChanged**()<br>Answer if the domain data changed since the last save/open |
| void | **DeleteRelations**(java.lang.String ConceptID, java.lang.String ResourceID)<br>Delete the relations in all domains between ConceptID and ResourceID |
| java.lang.String | **GetConceptName**(java.lang.String ConceptID)<br>Answer the name (local) name of the Concept identified by the String ConceptID |
| java.lang.String | **GetDomainForConcept**(java.lang.String aConceptID)<br>Answer the name of the domain the concept aConceptID belongs to |
| java.util.List | **GetDomainsForResource**(java.lang.String aResourceID)<br>Answer the names of the domains the resource aResourceID is used in |
| java.lang.String[] | **GetEquivalentConcepts**(java.lang.String aConceptID)<br>Give the concepts that are marked equivalent to the concept aConceptID |
| java.lang.String | **getPropertyValue**(java.lang.String aProperty)<br>Get the value for the property aProperty |
| java.lang.String[] | **GetRelatedConceptsForConcept**(java.lang.String ConceptID)<br>Answer an array of concept identifiers of all concepts which are related to the concept identified by the String ConceptID. |
| java.lang.String[] | **GetRelatedConceptsForResource**(java.lang.String ResourceID)<br>Answer an array of concept identifiers of all concepts which are related to the resource identified by the String ResourceID. |
| java.lang.String[] | **GetRelatedResourcesForConcept**(java.lang.String ConceptID)<br>Answer an array of resource identifiers of all resources which are related to the concept identified by the String ConceptID. |
| java.util.List | **GetRelationsForConcept**(java.lang.String aConceptID)<br>Get all relation structures for aConceptID |

| | |
|---|---|
| java.util.List | **GetRelationsForResource**(java.lang.String ResourceID)<br>Get all relation structures for ResourceID |
| void | **resize**()<br>resize the dmm interface upon resizing the OntoAIMS application |
| void | **ResourceDeleted**(java.lang.String ResourceID)<br>Method called by the Resource Management Module (RMM) when a resource is deleted from the resource pool. |
| void | **ResourceIDChanged**(java.lang.String OldID, java.lang.String NewID)<br>Notification from RMT that the identifier of a resource has changed from OldID to NewID |
| boolean | **ResourceUsed**(java.lang.String ResourceID)<br>Answer whether a resource from the resource pool, identified by the String ResourceID is used in a domain model. |
| void | **saveAll**()<br>Request from main program to save open domains to disk |
| void | **saveProperties**()<br>Save all open properties to disk |
| java.lang.String[] | **SearchConceptsForKeyword**(java.lang.String Keyword)<br>Return all concepts where the concept name equals the string Keyword |
| void | **setProperty**(java.lang.String aProperty, java.lang.String aValue)<br>Set the value of property aProperty to aValue |
| java.lang.String | **ShowPopup**(java.lang.String ResourceID)<br>Show a popup dialog to create a relation between a resource and a concept. |

### 7.2.3 RMT communication interface

public interface **RMTInterface**

*Interface to make the Resource Management Tool (RMT) accessible to other OntoAims Modules. It provides methods for accessing resource information, and methods to keep the resource pool consistent with the information in the other modules.*

Company: Eindhoven University of Technology

**Version:**

1.5

**Author:**

Johan Hubens., Jurgen Rijsdijk

---

## Method Summary

| | |
|---|---|
| boolean | **AddResource**(OAResource resX)<br>add resource to pool |
| void | **ConceptIsDeleted**(java.lang.String ConceptID)<br>Register the deletion of a concept in a domain in the domain editor module. |
| void | **ConceptNameChanged**(java.lang.String oldName, java.lang.String newName)<br>message that the name (id) of a concept had changed |
| void | **ConceptResourceRelationIsDeleted**(java.lang.String ConceptID, java.lang.String ResourceID)<br>Register the deletion of a concept - resource relation in the domain editor. |
| OAResource | **CreateResource**(java.lang.String URI)<br>create empty resource |

| | |
|---|---|
| boolean | **dataChanged**()<br>ask if the resourcepool has changed since last save to disk / file |
| OAResource | **GetResource**(java.lang.String URI)<br>get full resource from resource pool |
| java.lang.String | **GetResourceDescription**(java.lang.String ResourceID)<br>Answer a desription of the resource identified by the String ResourceID. |
| java.lang.String | **GetResourceIDFromPopup**(java.lang.String ConceptID)<br>Show a (popup) dialog to search for a resource from one of the other OntoAIMS modules. |
| java.lang.String[] | **GetResourceKeywords**(java.lang.String ResourceID)<br>Answer the keywords for the specified resource (id). |
| java.lang.String | **GetResourceName**(java.lang.String ResourceID)<br>Answer the name of the resource identified by the String ResourceID. |
| java.lang.String[] | **GetTopRatedResources**(java.lang.String ConceptID, int length)<br>Answer an array of resource identifiers which are most likely to be related to the concept identified by the String ConceptID. |
| boolean | **ResourceExists**(java.lang.String ResourceID)<br>Answer whether a resource identified by the String ResourceID exists. |
| void | **saveAll**()<br>request to save resources to disk / file |
| void | **ShowResourceInfoPopUp**(java.lang.String ResourceID)<br>Show a (popup) dialog presenting the metadata for the specified resource (id). |

### 7.2.4 Dictionary communication interface

public interface **DictionaryInterface**

Title: Dictionary Interface
Description: Interface to Dictionary class MyDictionary
Copyright: Copyright (c) 2004
Company: Eindhoven University of Technology

**Version:**

1.0

**Author:**

Jurgen Rijsdijk

---

| Method Summary | |
|---|---|
| boolean | **DictionaryAvailable**()<br>check if dictionary is installed & working |
| java.lang.String | **FindWord**(java.lang.String matchWord)<br>find if matchWord is in wordnet |
| java.util.ArrayList | **GetSynonyms_ar**(java.lang.String baseWord)<br>Get sysnonyms for word "baseWord" |
| java.util.ArrayList | **GetSynonyms_ar**(java.lang.String baseWord, int limit)<br>Get sysnonyms for word "baseWord" |
| java.lang.String[] | **GetSynonyms**(java.lang.String baseWord)<br>Get sysnonyms for word "baseWord" |

## *Appendix 8: RMT Search example*

The query a user enters to search for resources undergoes many steps in the transformation to search results. The following table shows an example of the data in the steps of a search.

| Query type | Step or query contents |
|---|---|
| User query | "an automobile or racecar" |
| | Step filer:<br>• Word "an" is shorter than 3 characters<br>• Word "or" is a common word |
| Filtered query | "automobile racecar" |
| | Step extend:<br>• "automobile" is not found as a domain concept<br>• "racecar" is found as domain concept 'racecar' in domain 'vehicles'<br>• Found synonyms for "automobile" are "car", "auto", "machine" and "motorcar"<br>• Found no synonyms for "racecar" |
| Extended query | • Set 1: "automobile"<br>o Concept =<br>o Synonyms = car,auto,machine,motorcar<br>• Set 2: "racecar"<br>o Concept = http://wwwis.win.tue.nl/aims/domain/Vehicles#racecar<br>o Synonyms = |
| | Step get resources |
| Q2 (extended query) + Matching Resources | • Set 0: "automobile racecar"<br>o Resources = 2<br>• Set 1: "automobile"<br>o Concept =<br>o Synonyms = car,auto,machine,motorcar<br>o Resources = 8<br>• Set 2: "racecar"<br>o Concept = http://wwwis.win.tue.nl/aims/domain/Vehicles#racecar<br>o Synonyms =<br>o Resources = 2 |
| | Step Cluster |
| Clusters | Main cluster: "automobile racecar", 2 resources<br>• Sub cluster: "automobile", 21 resources<br>o Sub cluster: "Domain sumo", 5 resources<br>o Sub cluster: "Domain cartoon", 9 resources<br>o Sub cluster: "Domain Vehicles", 8 resources<br>o Sub cluster: "Concept Amphibious_vehicle", 2 resources<br>o Sub cluster: "Concept Sportscar", 4 resources<br>o Sub cluster: "Source wikipedia", 5 resources<br>o Sub cluster: "Format reader", 2 resources<br>o Sub cluster: "Type Hypertext document", 5 resources<br>o Sub cluster: "Learning type slide", 5 resources<br>• Sub cluster: "racecar", 2 resources<br>o Sub cluster: "Domain: Vehicles", 2 resources<br>o Sub cluster: "Concept: Sportscar", 2 resources<br>• Sub cluster: "Domain: Vehicles", 2 resources<br>• Sub cluster: "Concept: Sportscar", 2 resources |

## *Appendix 9: OntoAIMS screenshots*

The Resource Management Tool is part of the OntoAIMS authoring environment, it is one of the three main components (DMT, CST, RMT). These three components are organized by a *tab structure*, where each tab provides access to a tool of the authoring environment. The main screen of the resource editor is shown below (2x):

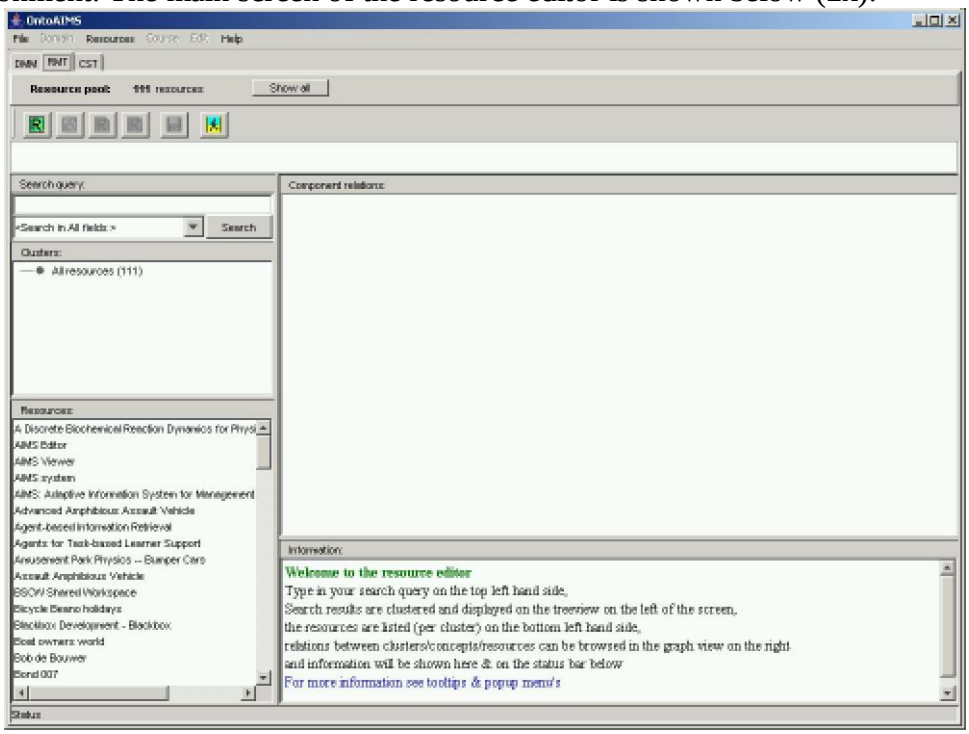


**Figure A 12, RMT main screen - on start**

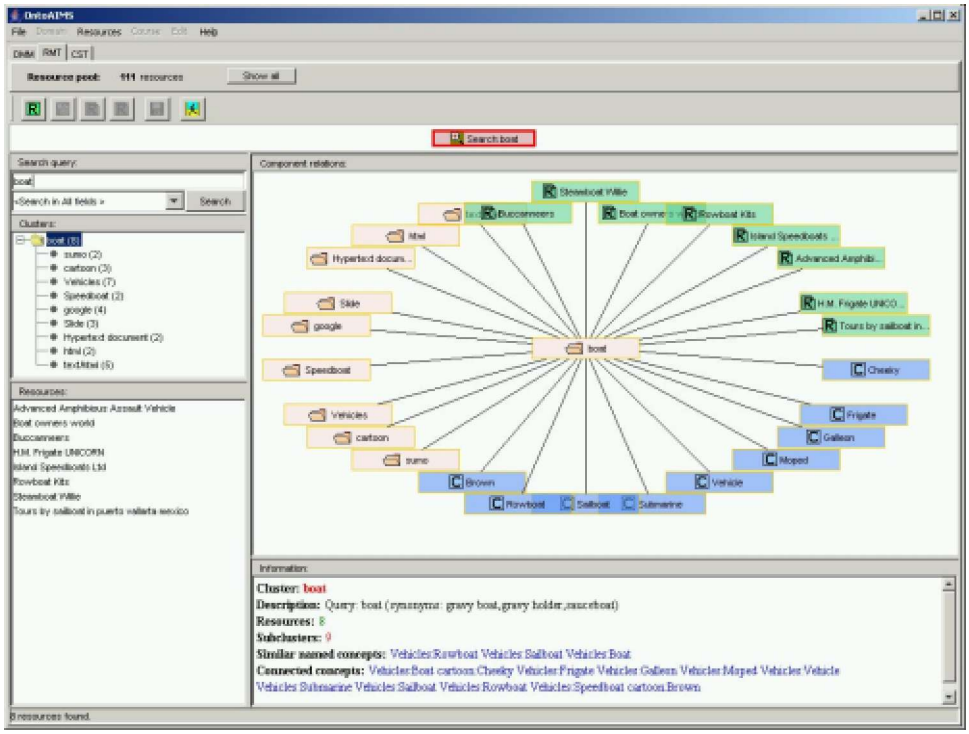


**Figure A 13, RMT main screen - after search**

The main window of the resource management tool (RMT) consists of eight parts:

- A button bar for quick access to management functions.
- A history bar that displays the recent selections and searches of the author
- A search form where the author can formulate a search request.
- A hierarchical navigation pane ('tree'), displaying the hierarchical structure of the clustering of search results.
- A list, which contains the (sub-) set of resources from the search results that belong to the currently selected cluster.
- A navigation pane ('graph'), visualising the relations (depth=1) of the currently selected object (resource, concept or cluster).
- A description pane providing a description of the selected object (resource, concept or cluster).
- A status bar to inform the author.

The combination of these user interface elements allows the author (user) to navigate the resources and related data structures (clusters & relations to domain concepts) When a selection changes in any of the GUI elements the other elements change accordingly.

To access the authoring features in the user interface there are several menus.
All the menus are partly enabled or disabled depending on whether there is a resource selected in the resource list and the graph.
There is a central menu:

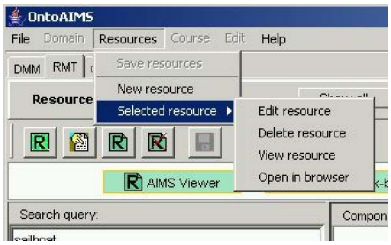


**Figure A 14, central menu enabled**


**Figure A 15, central menu disabled**

There is a popup menu in the graph

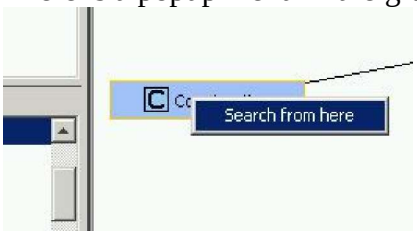


**Figure A 16, graph menu on concept or cluster**

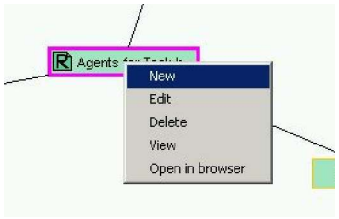


**Figure A 17, graph menu on resource**


**Figure A 18, graph menu on nothing**

There is a popup menu in the resource list.

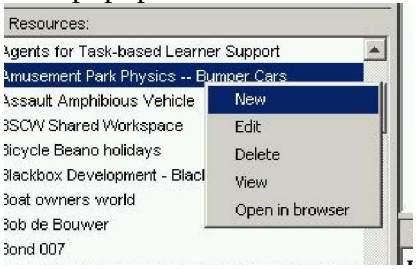


**Figure A 19, resource list menu enabled**


**Figure A 20, resource list menu on nothing**

To create, edit or view a resource there is a dialog screen to display the resource metadata. The resource metadata is more than a few items and therefore divided into several groups. These groups are defined in 3.3.3 OntoAIMS Resource metadata. There is a tab for each of these groups in the new/edit/view dialog.

3 examples of the new/edit/view dialogs (in different states):

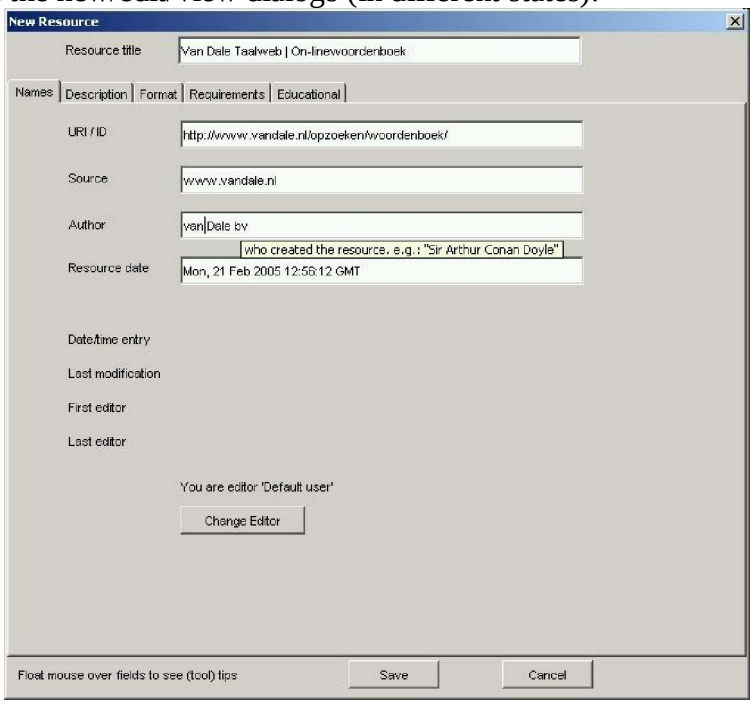


**Figure A 21, new resource dialog – first tab**

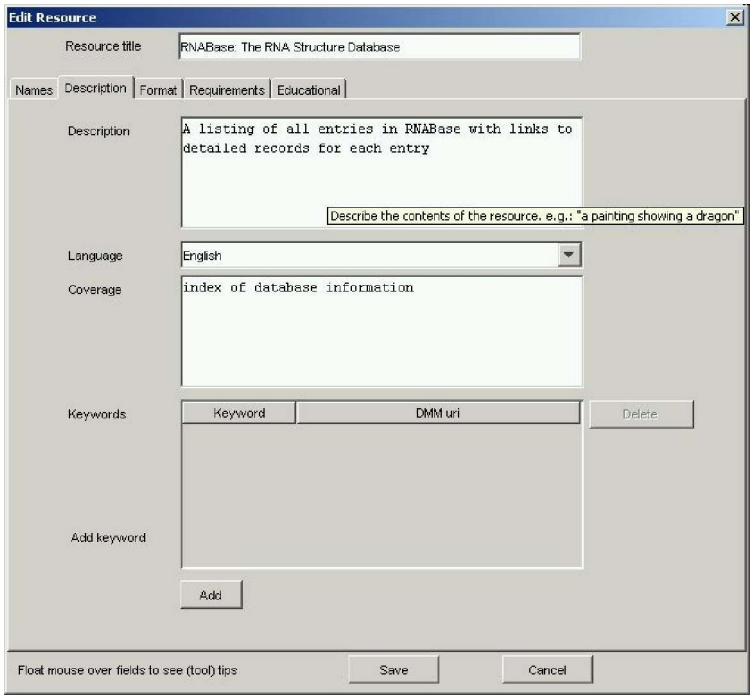


**Figure A 22, edit resource dialog, second tab**

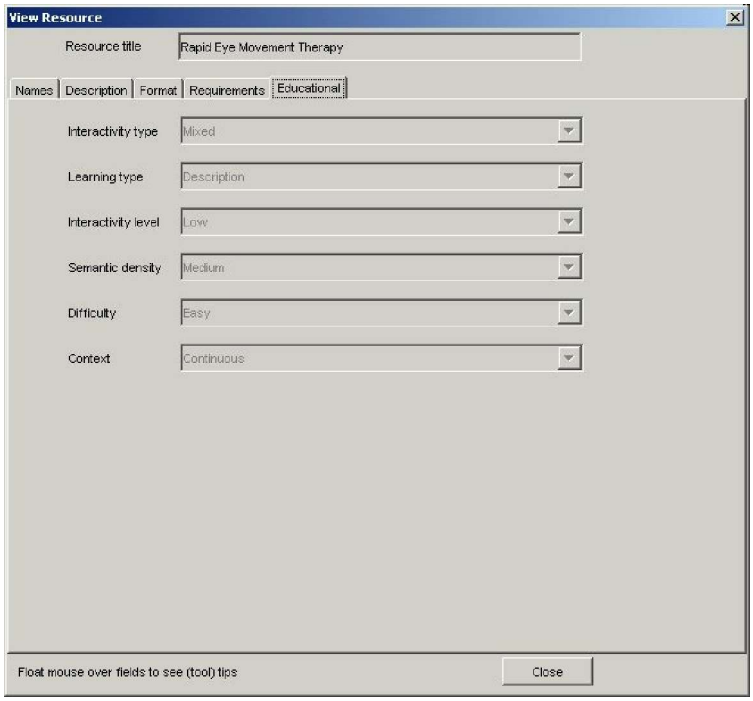


**Figure A 23, view resource dialog - fourth tab**

The new/edit/view dialog also has some support dialogs for specific tasks. When the author requests the new/edit/view dialog to input a new resource he is first asked for the URI of that new resource using this dialog:


**Figure A 24, create resource dialog**

This enables the system to support the author in entering metadata:

- The system can check for the existence of the URI in the resource pool thus preventing double entries
- Some information about the resource can be guessed by the system depending on the type of the URI.
- Some information about the resource may be retrieved from the actual resource (for instance creation time and type)[28].

[28] This is a very limited feature and might be improved upon in future work.

Other support dialogs for the new/edit/view dialog are “create requirement” and “create relation”.

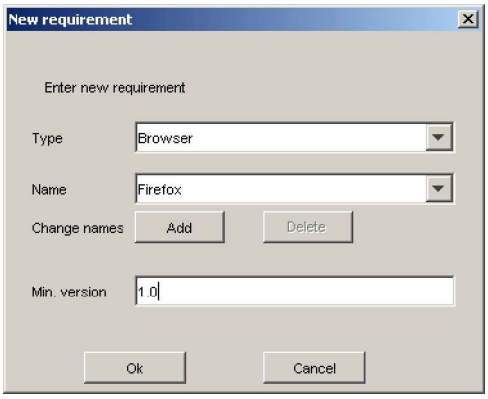


**Figure A 25, create requirement dialog**

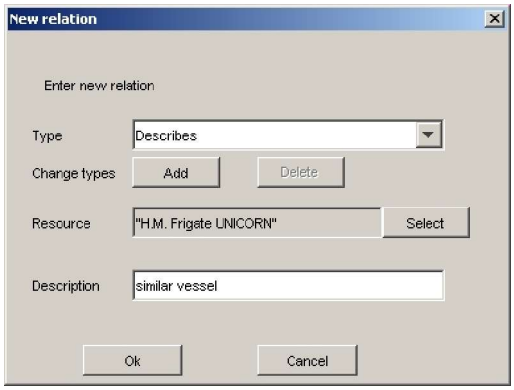


**Figure A 26, create relation dialog**

To inform the author of inconsistencies in data or warn about the consequences of deleting items error and information dialogs are shown.

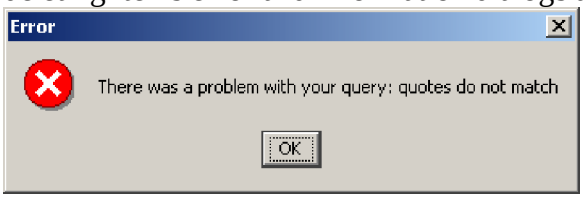


**Figure A 27, error dialog**


**Figure A 28, information dialog**

To use (domain) concepts in the RMT the DMT provides this dialog to select a concept:

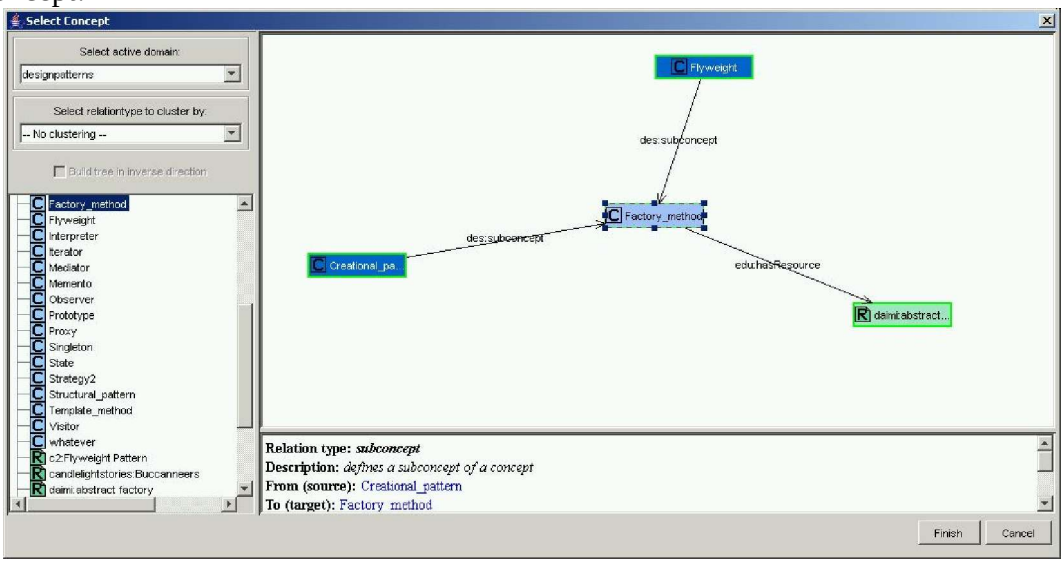


**Figure A 29, DMT select concept dialog**

To search for and use resources in the DMT o the CST the RMT provides the following dialog to the DMT and the CST:

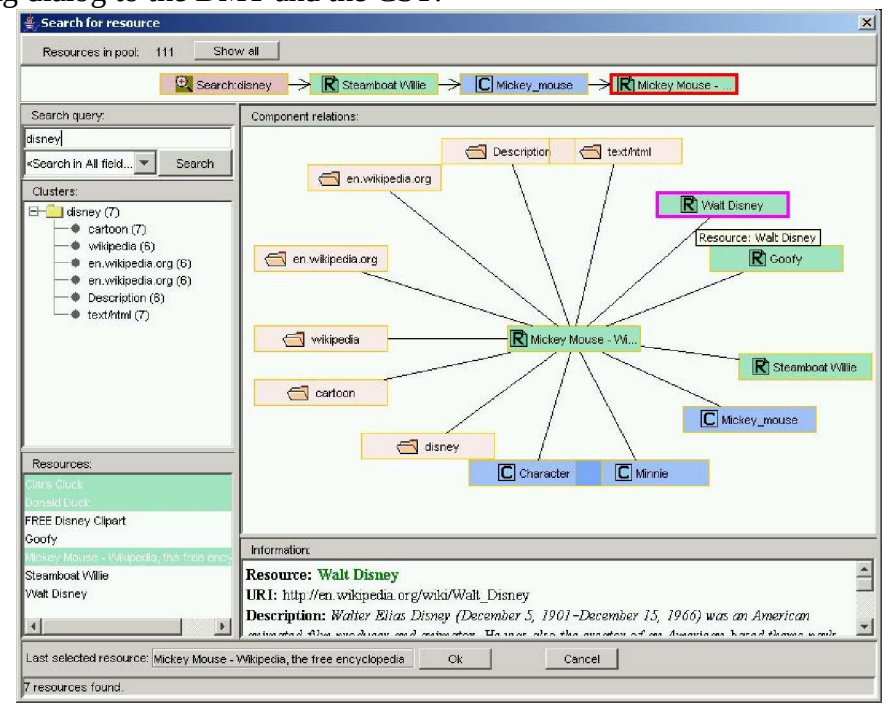


**Figure A 30, RMT resource search dialog**

This figure also shows that when the author hovers his/her mouse cursor over an object in the graph the related objects in the resource list are highlighted.

The same relation visualization is created from the mouse over resource list to highlighting elements in the graph as shown in the next figure:

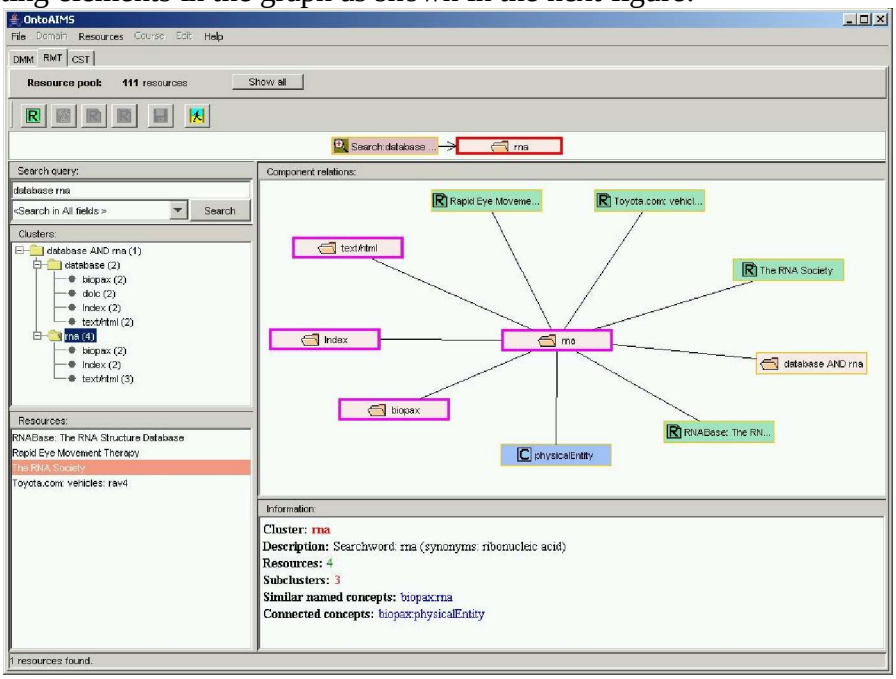


**Figure A 31, RMT main screen with relationship highlights**