Flexibility of Content for Organisational Learning
- A Topic Map Approach -

Stefan Oppl

May 22, 2007

'When I use a word,' Humpty Dumpty said in rather a scornful tone, 'it means just what I choose it to mean - neither more nor less.'
'The question is,' said Alice, 'wether you can make words mean so many different things.'
'The question is,' said Humpty Dumpty, 'which is to be master - that's all.'

Lewis Carroll in 'Through the Looking Glass' [Car71]

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Abschlussarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

DI Stefan Oppl

Contents

1 Introduction **1**
- 1.1 Motivation 1
- 1.2 Specification of Subgoals 2
- 1.3 Structure of this Work 4
  - 1.3.1 Part 1 - Requirements Gathering 5
  - 1.3.2 Part 2 - Refinement & Representation 5
  - 1.3.3 Part 3 - System Design & Implementation 6
  - 1.3.4 Part 4 - Evaluation 7

I Requirements Gathering **11**

2 Requirements from Organisational Learning **13**
- 2.1 Relevant Content Types 15
- 2.2 Requirements on Data Representation 16

3 Experiences from eLearning **19**
- 3.1 Decomposition of Content 20
- 3.2 Collaborative Learning through Communication 21
- 3.3 Learning Support using Process Models 21
- 3.4 Requirements on Data Representation 22

II Refinement & Representation **25**

4 Structured Refinement of Content Types **27**
- 4.1 Process 29
  - 4.1.1 Check against other Reference Models 36

4.1.2 Summary 45
4.2 Content 46
4.3 Communication 48
4.4 Meta-Data 49
4.5 Inter-model Aspects 49
4.5.1 Extensibility 51

5 Data Representation Concepts 53
5.1 Topic Maps 55
5.2 RDF & OWL 57
5.2.1 RDF & OWL Overview 57
5.2.2 Towards integration with Topic Maps 58

III System Design & Implementation 61

6 Overview 63
6.1 Technical Constraints for Scholion-Integration 64

7 Topic Map Engine 67
7.1 Persistency 68

8 Representation of OL Content 73
8.1 Mapping Content Models to Topic Maps 74
8.2 OL Management Layer 76
8.3 Scholion Data Importer 79

IV Evaluation 81

9 Content Structure Visualization 83

10 Evaluation Design 87
10.1 Formal Tests 89
10.2 User Evaluation 91

11 Evaluation & Results 93
11.1 Formal tests 93

11.2 User evaluation 96

12 Conclusions **99**
12.1 On the Use of Topic Maps 100
12.2 On relevant Schools of Knowledge Management 101
12.3 On Directions for further Development 102

References **104**

Appendix **114**

A Approaches to Organisational Learning **117**
A.1 Concepts focusing on the Learning Process 118
A.1.1 March & Olsen 118
A.1.2 Argyris & Schön 118
A.1.3 Huber 119
A.1.4 SECI-Model (Nonaka, Takeuchi & Krogh) 119
A.1.5 The Fifth Discipline (Senge) 120
A.1.6 Kim 120
A.1.7 Stoiber 121
A.1.8 ENRICH (Mulholland et al.) 121
A.1.9 Theory U (Scharmer, Senge, Jaworski & Flowers) 122
A.1.10 Knowledge Lifecycle (Firestone & McElroy) 122
A.1.11 Value Networks (Allee) 122
A.2 Concepts focusing on Objects of Learning 123
A.2.1 Stein 123
A.2.2 Abecker & van Elst 124
A.2.3 Eulgem 125
A.2.4 Linger & Burstein 125
A.2.5 Ramesh 126
A.2.6 Le, Lamontagne & Nguyen 126
A.2.7 Wargitsch & Wewers 127

B ISO Topic Map Details **129**
B.1 Topics . . . 129
B.1.1 Topic Names & Variants . . . 130
B.2 Associations . . . 131
B.2.1 Association Roles . . . 132
B.3 Occurrences . . . 132
B.4 Further Building Blocks . . . 133
B.4.1 Scope . . . 134
B.4.2 Meta-Elements - Types . . . 134
B.4.3 Reification . . . 135
B.4.4 Merging . . . 136
B.4.5 Subject Identifiers & Locators . . . 136

C Topic Map Engine **137**
C.1 Package ce.tm4scholion.tm . . . 137
C.1.1 Class Association . . . 138
C.1.2 Class AssociationRole . . . 144
C.1.3 Class Manager . . . 147
C.1.4 Class Manager.RoleTopic . . . 164
C.1.5 Class Occurrence . . . 165
C.1.6 Class Reifiable . . . 170
C.1.7 Class Scope . . . 172
C.1.8 Class Statement . . . 176
C.1.9 Class Topic . . . 178
C.1.10 Class TopicMap . . . 188
C.1.11 Class TopicMapConstruct . . . 193
C.1.12 Class TopicName . . . 196
C.1.13 Class Utils . . . 202
C.1.14 Class Variant . . . 206
C.2 Package ce.tm4scholion.tm.persistency . . . 210
C.2.1 Interface TMPersistency . . . 211

D OL Content Models **213**
D.1 Package ce.tm4scholion.metamodel . . . 214
D.1.1 Class Element . . . 214
D.1.2 Class Manager . . . 218

D.1.3 Class Manager.RoleElementCombination 225
D.2 Package ce.tm4scholion.metamodel.common 226
D.2.1 Class Course 227
D.2.2 Class Manager 229
D.2.3 Class Subject 230
D.3 Package ce.tm4scholion.metamodel.learning 232
D.3.1 Class Block 232
D.3.2 Class LearningUnit 235
D.3.3 Class Manager 236
D.4 Package ce.tm4scholion.metamodel.communication 237
D.4.1 Class Entry 238
D.4.2 Class EntryContainer 239
D.4.3 Class Manager 241
D.5 Package ce.tm4scholion.metamodel.communication.chat 244
D.5.1 Class Chat 244
D.5.2 Class ChatEntry 246
D.5.3 Class Chatroom 247
D.6 Package ce.tm4scholion.metamodel.communication.forum 249
D.6.1 Class Discussion 249
D.6.2 Class DiscussionEntry 250
D.6.3 Class DiscussionTopic 251
D.6.4 Class Forum 253
D.7 Package ce.tm4scholion.metamodel.communication.infoboard . . 254
D.7.1 Class Infoboard 254
D.7.2 Class InfoboardEntry 255

List of Figures

1.1 Relationships between Goals 4
1.2 Overview of Structure of this Work 9

3.1 Learning support with process models 22

4.1 Information Types for OL 28
4.2 The mediational structure of an activity system (taken from [Eng87]) 32
4.3 Conceptual model for procedural content - Meta-level 34
4.4 Conceptual framework for procedural content - A detailed view on model elements 37
4.5 Modeling approaches for relevant aspects of business information systems 38
4.6 WFMC Basic Process Definition Meta-model (taken from [Wor95]) 38
4.7 Meta Model for Task Modeling (adapted from [vW01]) 40
4.8 Meta Model for Process Modeling (taken from [SAJ+02]) 43
4.9 Conceptual Content Framework 48
4.10 Conceptual Communication Framework 50

5.1 Topic Maps - Basic Elements (taken from [ISO06a]) 56
5.2 Topic Maps - Structural Overview 56

6.1 Architectural Overview 64

7.1 Topic Map Engine - Class Hierarchy Overview 69
7.2 Topic Map Engine - Class Property & Association Overview 71

8.1 Representation of Scholion Content using TopicMap concepts . . 77
8.2 Structure of OL Management Layer 78

9.1 Example of Content Structure Visualisation 84

11.1 Visualisation of complete topic map structure 94

B.1 The Semiotic Thetrahedon (adapted from [FHL+98]) 130
B.2 Topic Maps - Topic (taken from [ISO06a]) 130
B.3 Topic Maps - Topic Naming 131
B.4 Topic Maps - Association (taken from [ISO06a]) 132
B.5 Topic Maps - Occurrences 133
B.6 Topic Maps - Full Overview 133
B.7 Topic Maps - Example for super/subtyping and instancing 135

D.1 Overview of currently implemented content model structure . . . 213

List of Tables

4.1 Mapping of the WfMC Reference Model to the Conceptual Model 39
4.2 Mapping of the Task Modeling Meta Model to the Conceptual Model 41
4.3 Mapping of the Process Modeling Comparison Framework to the Conceptual Model 44

Chapter 1

Introduction

'Where shall I begin, please your Majesty?' He asked.
*'**Begin at the beginning**,' the King said, very gravely,*
'and go on till you come to the end: then stop.'

The White Rabbit & the King in [Car66]

The title of this work is *Flexibility of Content for Organisational Learning*. Consequently, the overall goal is to develop a means to flexibly represent and manage content for organisational learning.

1.1 Motivation

Many approaches to explain and/or define the process and the objects of organisational learning have been proposed in the last decades. Most of these approaches give detailed statements on the process of organisational learning or focus on the enablers of and obstacles within this process. However, when it comes to the objects of organisational learning, they either remain unspecific (e.g. Argyris & Schön [AS78] or March & Olsen [MO82]) or explicitly argue for being open in these terms (e.g. Kim [Kim93] with his statements on mental models or Firestone & McElroy [FM03b] with their definition of knowledge claims).

The ambiguity and uncertainty of *what to learn* in organisational learning remains uncritical as long as the frame of reference is a purely social one (in contrast to socio-technical systems). Considering organisations as social systems, organisational learning (OL) occurs directly among individuals. The main

concern of most OL approaches is to enable and aid the learning process itself, the objects of learning are of lower interest, as they are transfered directly on an interpersonal level.

Considering organisations as socio-technical systems, however, broadens the scope of possible carriers and facilitators for OL. Especially ICT-based tools to enable and support learning provide a promising infrastructure for OL. In the ICT-context, knowing the objects of learning becomes crucial. For ICT-supported learning, content has to be represented in data structures, which requires to have a defined data model. A data model can only be designed, if the data to be represented (and thus the objects of learning) are known.

As mentioned before, most OL approaches consider the objects of learning to be of arbitrary form and structure (e.g. see Firestone & McElroy [FM03b]). ICT-support of OL can only be enabled, if a form of representation is found, that allows flexible management of arbitrary information relevant in the context of OL.

1.2 Specification of Subgoals

The subgoals given in the following define the scope of research for this work. Each of them is put into the context of the global goal (giving a rationale in doing so). The tasks to reach a subgoal are defined for each of them. Figure 1.1 shows the relationships between the subgoals.

1. Identification of requirements on representation from OL
   Specifies the details on what flexibility in representation means from an OL point-of-view
   - Review of OL approaches regarding statements on how objects of learning have to be (re)presented
   - Extraction of requirements on representation
2. Identification of content types relevant for OL
   Identifies potential objects of learning OL and so provides examples for the need of flexible representation of content
   - Review of OL approaches regarding statements on relevant objects of learning

- Consolidation of statements and extraction of content types considered relevant for this work

3. Identification of requirements on content representation in ICT-supported learning
   Determines the concepts of content representation to enable individual learning in ICT-supported settings

   - Review of existing eLearning concepts regarding statements on how learning content is (re)presented
   - Review of an existing platform implementation regarding the technical requirements on representation

4. Finding a appropriate concept for representation
   Identifies an approach to data representation which matches the requirements identified above

   - Identification of a concept for representation based on the requirements
   - Development of transformations between content requirements and actual representation

5. Representation of different types of content
   Shows how different types of content can be represented meeting the defined requirements

   - Development of data models for each identified content type based on the requirements on data representation

6. Technical realisation of representation and management
   Brings together means and subject of representation and provides the actual result of this work

   - Design and implementation of the data management logic
   - Design and implementation of interfaces to access content

7. Checking the functionality and appropriateness of the developed concepts and implementations
   Shows that the proposed concepts work technically

- Formal software test of implemented components
- User evaluation of the implemented system

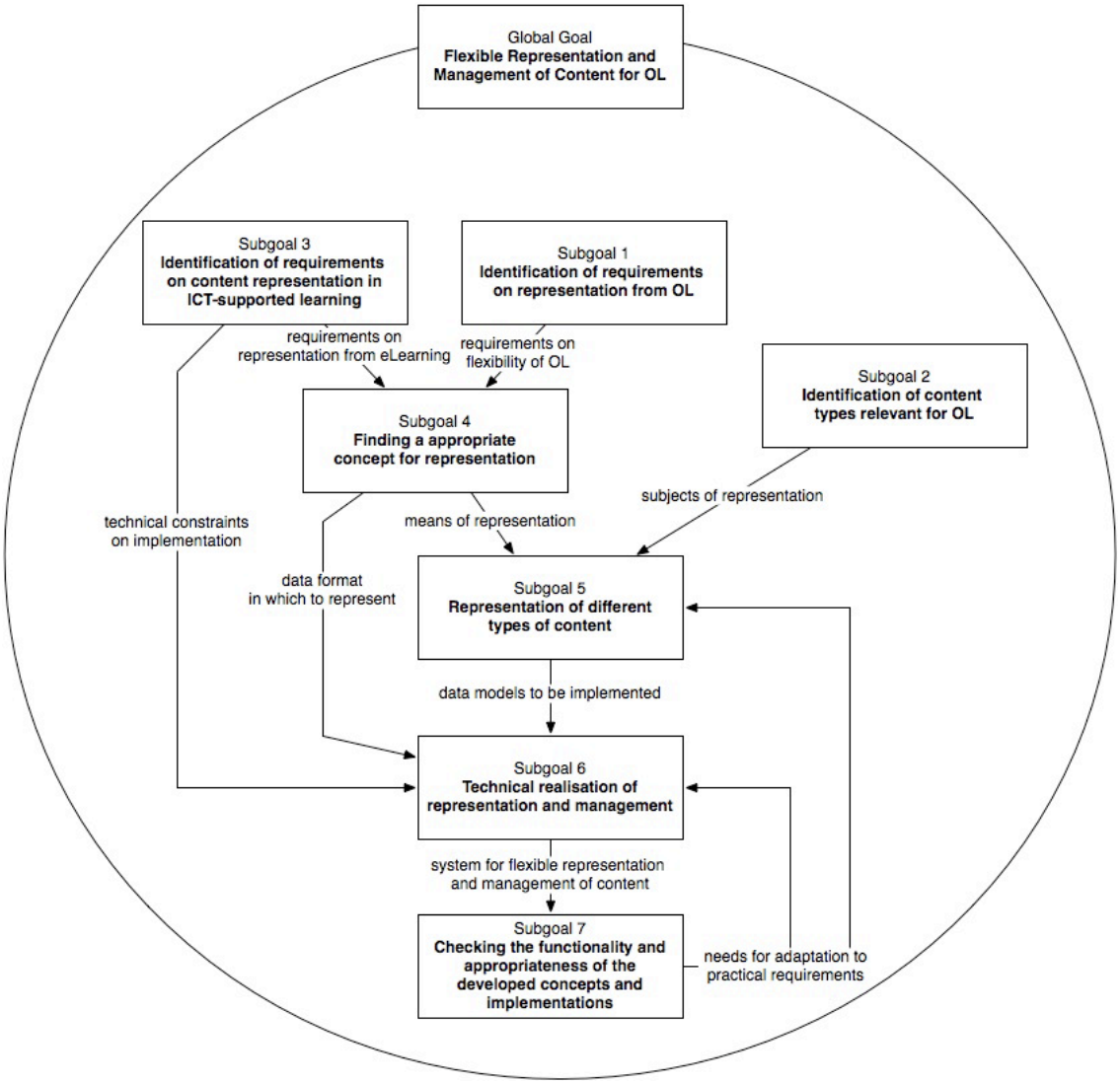

Figure 1.1: Relationships between Goals

1.3 Structure of this Work

In this section, the structure of this work is presented. Each chapter is assigned to specific subgoals.

At the beginning of each chapter, its goals are put into the context of the whole work and so set a frame of reference for the presented content. At the end of each chapter, recapitulation of the rsults points ot the contributions to the overall goal.

The descriptions given here already anticipate some results of the first chapters (in mentioning Topic Maps as an appropriate means of representation). In this way, a brief and structured overview of this work is presented here.

1.3.1 Part 1 - Requirements Gathering

Chapter 2 - Requirements from Organisational Learning In this chapter, the tasks of subgoals 1 and 2 are described. A literature study on organisational learning and organisational memories is carried out. The focus of this study is on statements of relevant types of organisational knowledge and requirements on representation. Annex A complements chapter 2 with the detailed results of the review. The results of this chapter are twofold. First, a list of requirements on representation derived from OL approaches is given. Second, a set of content types considered relevant is consolidated from the statements on the objects of learning in OL approaches.

Chapter 3 - Experiences from eLearning Chapter 3 focuses on the eLearning platform Scholion as a representative platform to be further developed. Scholion is used as an sample platform, on which the transition from eLearning to OL support is shown. The underlying concepts and their implications on representation of content are described. In addition, an approach using process visualizations to facilitate learning is reviewed, as the concept is considered relevant for OL. This chapter covers subgoal 3 and results in a set of requirements on (re)presentation from ICT-supported learning.

1.3.2 Part 2 - Refinement & Representation

Chapter 4 - Structured Refinement of Content Types In chapter 4 the identified content types are brought together with the requirement on fine-grain content decomposition set by Scholion. For each content type, a structured, block-oriented model is developed. The structure of the types already used in

Scholion are derived from the concepts applied there. The new content type 'work process' is developed from scratch based upon the conceptual framework of activity theory. This chapter covers subgoal 5 and presents models of every identified content type as results.

Chapter 5 - Data Representation Concepts This chapter sets out to identify and describe the actual means of data representation and thus covers subgoal 4. The technical requirements on the data representation concept are derived from the conceptual requirements identified in chapter 2 and 3. Based upon these requirements, two possible approaches are identified, of which topic maps are selected as the more suitable one. The basic concepts of topic maps are described and mapped to the requirements. Annex B complements this description with details on topic map constructs. The second approach, RDF/OWL, is then reviewed regarding possible complementing or conflicting concepts. The decision to use topic maps as the concept for content representation in this work is the result of this chapter.

1.3.3 Part 3 - System Design & Implementation

Chapter 6 - System Design Overview Chapter 6 gives an overview of the system's architecture and the components to be implemented. The demand for integration in an existing IT infrastructure leads to implementation constraints, which are also presented here. Chapter 6 thus contributes to reaching subgoal 7 by setting the frame of reference for implementation. There are two results of this chapter: the top level architecture and the technical constraints for implementation.

Chapter 7 - Topic Map Engine The implementation of the topic map concepts for data representation is presented in chapter 7. It contributes to reaching subgoal 6 by providing the core, low-level data management component. Annex C provides implementation details of the topic map engine. In addition, means to store and retrieve topic maps are presented. The major output presented in this chapter is the topic map engine, which is the basis for all further implementations.

Chapter 8 - Representation of OL Content In this chapter, the content models developed in chapter 4 are broken down to actual topic map-based representations. The identification of common structures in the various models provides the foundation for concepts for consistent representation of arbitrary content. A meta-model representing the mapping of models to topic maps is defined and implemented for all content types. Annex D gives details on the implementation for these components. The resulting content management layer completes the work on reaching subgoal 6.

1.3.4 Part 4 - Evaluation

Chapter 9 - Content Structure Visualization An application for content structure visualization is designed and developed as a proof of concept in chapter 9. This application is also used for the detailed tests on functionality and applicability in actual usage scenarios. Based upon the GraphViz toolset, a visualizer is developed, that generates hyperlinked graphical representations of the content structure including links to exemplary content. This chapter contributes to reaching subgoal 7 by demonstrating the basic feasibility of the approach and providing the means of detailed evaluation.

Chapter 10 - Evaluation Design In chapter 10, the evaluation of the developed concepts and the system implementing them is planned. Evaluation is carried out on two levels. First, a technical evaluation regarding correct functionality is carried out. For this part, detailed tests and criteria to be met are specified. The second part checks feasibility and appropriateness of the implemented concepts in a user test. Both parts use exemplary content designed to cover all implemented concepts and functions. The result of this chapter is the evaluation design for this work including test cases and criteria. This contributes to reaching subgoal 7.

Chapter 11 - Evaluation & Results In chapter 11, the results of evaluation are described based upon the evaluation design given in the former chapter. The components to be refined or completed in future work are identified. The results of this chapter allow to reach subgoal 7.

Relationships among Chapters The chapters are related to each other basically along the goal structure given in figure 1.1. Figure 1.2 shows the relationships between the chapters based on the top level results.

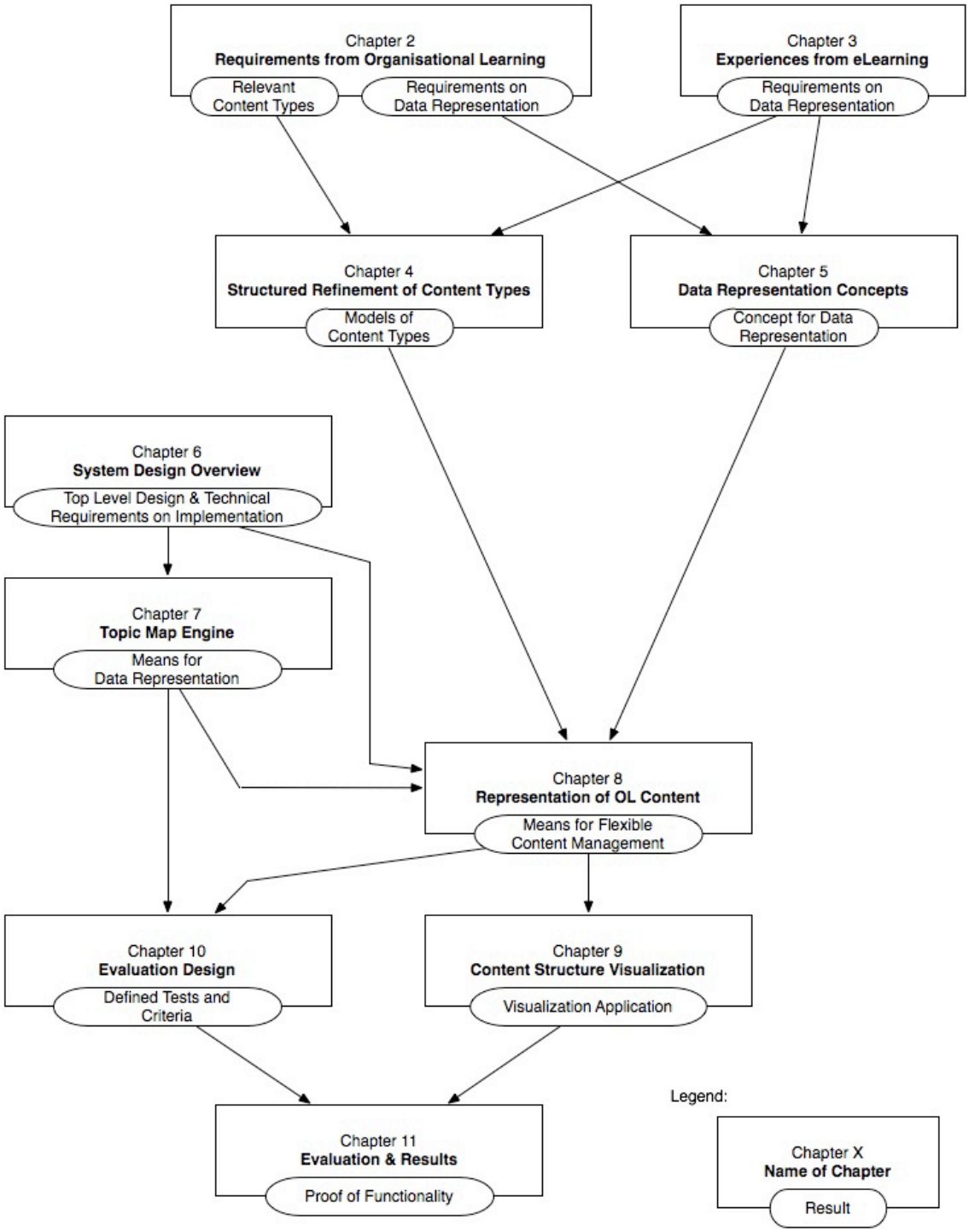

Figure 1.2: Overview of Structure of this Work

Part I

Requirements Gathering

Chapter 2

Requirements from Organisational Learning

The reality is that successful, and sustainable, adaptation is driven by distributed Knowledge Processing, characterized by free thinking workers whose self-organizing patterns create organizational knowledge in an atmosphere of openness in problem recognition, solution formulation, and solution evaluation.

Joseph M. Firestone and Mark W. McElroy in [FM03a]

Objectives of Chapter 2

In this chapter, approaches to explain and describe aspects of Organisational Learning are examined. There are two goals in this chapter. The first is to identify the types of content relevant for OL. The second goal is to derive requirements on data representation from the reviewed approaches. The latter is necessary as a basis for discussion of suitable forms of data representation. The types of content will be refined and mapped to the chosen form of representation to show feasibility of the approach in chapters 5 and 8.

For identification of the relevant types of learning and the requirements on data representation, existing work on organisational learning is reviewed. In OL, most approaches focus on one of the two following points of view:

Focus on the process of learning , that is *how* an organisation learns and how to facilitate learning. In this context, statements are also given on *what* to learn and *requirements* on (re)presentation for learning. Approaches focusing on these aspects are often referred to as *Organisational Learning Frameworks*

Focus on the objects of learning , that is *which* types of content are relevant for an organisation to store and distribute. Approaches focusing on this area of research are often also referred to as *Organisational Memories*.

Hardly any approach can be assigned to only one of these foci in all its aspects. Consequently, not difference is made between the categories as regards review. However, for a more structured overview, the reviewed approaches are sorted according to their main focus.

In her PhD-thesis [Sto03], Stoiber has reviewed frameworks for organisational learning and organisational memory concepts. Considering her work and the evolution of OL-research since then, her list has been extended by more recent approaches. These approaches have been reviewed regarding types of content and requirements on data representation. A detailed description of this review is given in annex A. In the following, only the results are presented. A more comprehensive overview of most of the presented OL- and OM-models is available in [Sto03].

The following approaches have been reviewed:

- Concepts focusing on the learning process
  - Cycle of Choice (March & Olsen)
  - Argyris & Schön
  - Huber
  - SECI-Model (Nonaka, Takeuchi & Krogh)
  - The Fifth Discipline (Senge)
  - Kim
  - Stoiber
  - ENRICH (Mulholland et al.)
  - Theory U (Scharmer, Senge, Jaworski & Flowers)
  - Knowledge Lifecycle (Firestone & McElroy)
  - Value Networks (Allee)

- Concepts focusing on the objects of learning
  - Stein
  - Abecker & van Elst
  - Eulgem
  - Linger & Burstein
  - Ramesh
  - Le, Lamontagne & Nguyen
  - Wargitsch & Wewers

From the studied OL frameworks it can be concluded, that organisational learning always involves or is primarily based on individual learning processes. Consequently, both, individual learning and bringing together the individuals for an organisational learning step, has to be supported to enable OL.

2.1 Relevant Content Types

Summarizing the results of sections A.1 and A.2 in annex A, the following information types are considered relevant. The references in brackets refer to the approaches which argue for the respective requirement.

- Structured, sequential representations of work (Stoiber, Stein, Abecker & van Elst, Wargitsch & Wewers)
- Networked, graphical representations of organisational structure (Stoiber, Allee, Le et al.)
- Textual representation of work- and organisation-relevant information (Mulholland et al., Abecker & van Elst, Eulgem, Linger & Burstein, Ramesh, Le et al., Wargitsch & Wewers)
- Prescriptive representation of (standardized) work processes (Stein)
- Narrative representation of information, communication (Abecker & van Elst, Wargitsch & Wewers)
- Information about processes within organisations (Argyris & Schön, Stoiber, Allee, Stein, Abecker & van Elst, Eulgem)
- Information about organisational structure (Stoiber, Allee, Stein, Abecker & van Elst, Eulgem)

- Conceptual, declarative understanding of real world phenomena (Nonaka & Takeuchi, Kim, Allee, Stein, Abecker & van Elst, Linger & Burstein, Ramesh, Le et al.)
- Descriptions of how to perform a task, know-how, rules, norms (Nonaka & Takeuchi, Kim, Stoiber, Allee, Stein, Abecker & van Elst, Linger & Burstein, Ramesh, Le et al.)
- Structural meta data about information objects (Abecker & van Elst)
- Domain-specific meta data about information objects (Abecker & van Elst, Ramesh)
- Background information that aids understanding of actual learning content (Le et al.)

Summing up, these statements lead to the following types of content considered relevant for OL in this work:

- Structured, graphical descriptions of work and its organisational context
- Textual descriptions of information relevant in an organisation
- Communication
- Meta-data to describe any of these

2.2 Requirements on Data Representation

The following requirements on content for OL can be derived from the reviewed approaches:

- Provide information that is relevant in the current context of work (March & Olsen, Mulholland et al., Linger & Burstein)
- Provide the actual work context to learning content (Mulholland et al., Firestone & McElroy)
- Provide information to discover and resolve mismatches between expected and observed behaviour (Argyris & Schön, Firestone & McElroy)

- Support the exchange of uncodified (implicit) information (Argyris & Schön, Huber, Nonaka & Takeuchi, Krogh, Senge)
- Provide manifold, media-rich representations of learning content (Huber)
- Provide means of efficient navigation and content access to avoid information overload (Huber)
- Provide means to distribute individual insight and findings (Kim, Krogh)

For selection of a means for representation, the following features are considered necessary. They are derived from the statements given above. Representation has to allow:

- representation of different content types
- storage of single content elements in different, arbitrary forms of (re)presentation
- adding individual information and remarks to content
- putting content elements into arbitrary relationship with each other
- attaching structural and domain-specific meta-data to content elements

Chapter 2: Summary

The requirements on data representation and the content types relevant for OL have been derived from an extensive review of OL approaches in this chapter. The former are used as a basis for selection of the actual form of data representation. The latter are refined (using the concepts presented in the following chapter) and mapped to the selected form of representation. This allows to represent content in each of the identified types and is necessary to show feasibility of the approach.

Chapter 3

Experiences from eLearning

The problem with even the best learnings is that without some kind of infrastructure for thinking things through and a way to codify the learnings in a coherent way, we often stay stuck in situational, fragmented, or opportunistic learning.

Daniel H. Kim in [Kim01]

Objectives of Chapter 3

The structuring of content for eLearning is reviewed in this chapter. This work sets out to extend the functionality of a concrete eLearning platform, Scholion [AS05a]. Using Scholion as a sample platform, this work shows, how the transition from eLearning to OL support might work. Thus, focus is on the concepts developed and implemented there. The results of this chapter will add to the basis for discussion of suitable representation concepts.

As stated before, organisational learning always involves individual learning processes. These can be supported with ICT-based tools, as research in the domain of *eLearning* has shown in the last years (e.g. cf. to [Aui03] for evaluation results of the platform extended in this work). Previous research in content engineering for eLearning ([AS05b], [AS05a]) argues for decomposition of learning content into semantically consistent blocks (e.g. examples, definitions or motivations). These blocks have to be provided in different forms of presentation

to support different learning situations. Knowledge transfer among all participants is facilitated by means of communication in the context of the learning content.

Another aspect of learning support has been adressed by Kienle, Herrmann et al. (e.g. [CHKM05][KH04b]) They have examined the role and potential of guiding process models in collaborative eLearning settings (which are also relevant for the support of OL, as identified in chapter 2).

3.1 Decomposition of Content

The basic building blocks for content presentation in Scholion are *Blocks* and *Learning Units*. Learning units correspond to the concept of learning objects in literature (e.g. [Sch05b], [Paw01] or [Pol03]). Learning objects are *self-contained* representations of a certain topic. They should be of *similar granularity*, e.g. each representing the equivalent of one lesson and should have a *standard structure* to provide a coherent appearance [Sch05b].

The internal structuring of learning units is done with *blocks*. A block is the basic unit of decomposition and representes a content element with a certain semantic connotation, which is explicitly defined in *block types*. These *block types* are didactically motivated and can be extended or altered domain-specifically. Examples of block types are *example*, *definition*, *motivation* or *code* (to give a domain-specific type from software-engineering domain).

Making use of the block type concept, didactic patterns can be specified, which provide support for consistent authoring of learning content. Learning units (and consequently blocks) can also be made available in different *levels of detail*. Depending on the learning setting, the learner can choose between LOD 1 (keywords, slide-oriented), LOD 2 (full text, textbook-oriented) and LOD 3 (additional or background information). Both, learning units and blocks, are complemented by *structural and domain specific meta data*, such as author, creation date or keywords.

The concepts presented so far are generic means for content structuring and augmentation, that are applied on a global, not learner-specific level. Scholion furthermore uses concepts for individualization of content. *Views* can be considered individual overlays for content that contain annotations, mark ups or links. Views enable learners to personalize their learning content by integrat-

ing individual semantic information into learning content or linking to related information.

3.2 Collaborative Learning through Communication

Communication support is considered very relevant for eLearning in order to enable group work like in traditional, presence-based learning settings [Aui03] - this is especially true for OL support (cf. chapter 2).

Scholion provides means for synchronous and asynchronous communication, unidirectional from teacher to learner as well as bidirectional among all participants. The *infoboard* is a means to publish information on a virtual bulletin board (asynchronous, unidirectional). For bidirectional communication Scholion provides *forums* (asynchronous) and *chats* as well as an *instant messenger* (synchronous).

Furthermore it is possible to share views among members of a learning group or to provide access to certain views for all participants. This allows content-based asynchronous collaboration among learners, where communication is always linked to the actual learning context.

3.3 Learning Support using Process Models

In recent work, Carell et al. ([CHKM05], [KH04b], [KH04a]) have examined the effects of visual guidance through the learning process in eLearning. They have integrated visual models of the learning tasks into an eLearning plattform and have linked tasks to learning content and communication features. Their approach enables learning in the context a university studies setting. The concepts applied there are not restricted to usage in institutionalized learning. Although not yet evaluated, they seem to be applicable in OL support, too. Providing the work context of learning is a requirement for successful OL support.

Figure 3.1 shows, how process support is implemented. Learners are presented a visual representation of the tasks necessary to complete the learning unit. Tasks are decomposed to a level, on which they are comprehensible for

users, so that they can start learning immediately and put their skills into the context of the overall task.

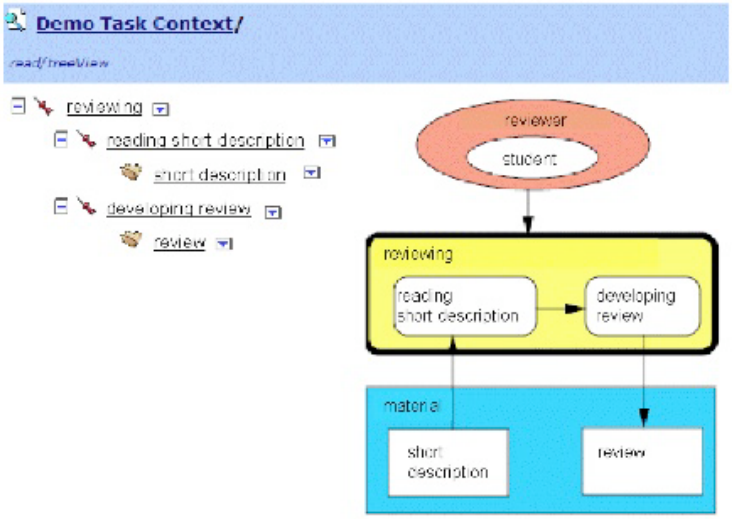

Figure 3.1: Learning support with process models

The results of the study they conducted seem promising: They state that *the usage of graphical process models during the preparation of the collaboration can lead to more knowledge exchange and integration, as well as commitments concerning the collaborative learning process, and a more intensive and collaborative usage of the CSCL-system* [CHKM05]. These results encourage to follow similar concepts in extending Scholion with features for organisational learning.

3.4 Requirements on Data Representation

All the concepts presented here rely upon fine-grain decomposition of content. Learning content is not treated as one single document but is decomposed into semantically consistent learning blocks. Communication content is also processed on the level of single contributions to discussions. Consequently, the level of granularity in process visualizations is also brought down to single tasks.

Content of any kind is assembled again by specifying relationships among the learning blocks. In most cases these relationships are hierarchical or sequential.

However, there are some cases within content types but especially on a global, inter-type level - where relationships with open semantics are needed (e.g. when linking learning content with communication, an 'refines' or 'discussed in' relationship type makes sense).

For data representation, this leads to a demand for a concept that allows

- arbitrarily definable blocks of content
- arbitrarily definable association of blocks of any type
- arbitrarily definable forms of representation of content blocks

Chapter 3: Summary

This chapter completes the basis for discussion of a suitable form of data representation. While in the former chapter, focus was on the content dimension, this chapter has focused on the didactic dimension. The requirements of both chapters lead to the selection of a data representation concept in the next part.

Part II

Refinement & Representation

Chapter 4

Structured Refinement of Content Types

The widespread attitude of researchers in the information system field to create Yet Another Modelling Approach (the 'YAMA Syndrome'), without addressing the question of its justification and its scientific value added sufficiently, is another one. The variety of existing modelling languages and dialects is therefore unnecessarily large. Apart from a lot of subjectivity, a lot of arguments like 'apples taste better than oranges', there are also some good reasons for differences, as far as substantial aspects are concerned. Different designers and users of an information system may view the world differently.

E.D. Falkenberg et al. in the FRISCO Report [FHL+98]

Objectives of Chapter 4

In this chapter, the information types identified in the last part are refined to structured, block-oriented models of content representation. This is necessary to meet the requirements of eLearning-support, where decomposition of content is a fundamental concept. The models developed here are used for the specification of content representation.

For structured data representation, specification of data models is necessary.

In the former part, three content types have been identified. The refinement into object-oriented data models is carried out for these three here. However, also other content types can be relevant dependent on the application domain. Those have to be refined in the same way to be representable using this approach. The three types examined here are:

- Work-Process (procedural information)
- Learning-Content (descriptive information)
- Communication (narrative information)

These three have to been intertwined (like it is already the case with learning content and communication in *Scholion*). A fourth content type - meta-data - is used to provide further context for the respective element (not in the sense of organisational context but as in the sense of an element life-cycle documentation) (see figure 4.1). While learning-content, communication and meta-data are basically adopted from the existing concepts in Scholion, work process has to be developed from scratch based on previous research in this area.

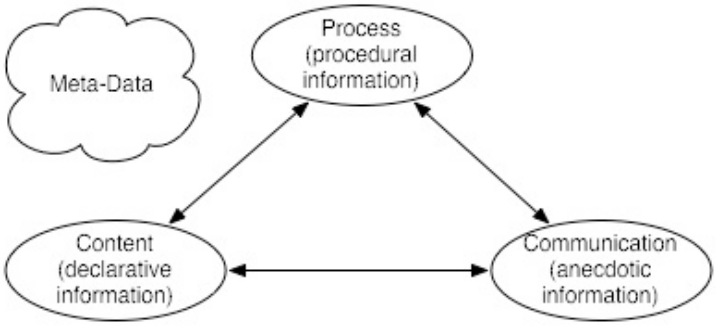

Figure 4.1: Information Types for OL

The developed models together form a conceptual framework for OL content representation. This conceptual framework follows a multi-level approach of model representation. It contains means to represent models as well as model instances (e.g. cf. to section 4.1 for the part on representation of procedural information and models).

4.1 Process

In this section, the conceptual model for process representation is derived from foundations in *Cultural Historical Activity Theory* (*CHAT*). It is then contrasted with several other reference-models in the process- and task-modeling area to cross-check expressiveness and unambiguousness of possible mappings to those approaches.

The conceptual model is based on Leontiev's *Activity Theory* [Leo78]. In its further developed and extended form ([Eng87], [CE93]), *Cultural Historical Activity Theory* is capable of explaining how people work together. CHAT is widely acknowledged as a appropriate foundation for research in the fields of HCI (e.g. [KNM99], [Nar96]), Software Engineering [DR97] or Organisational Learning (e.g. [VK00], [Eng05]). For an comprehensive introduction to Activity Theory, with an emphasis on HCI research, see [BB03].

Following the introductions to Activity Theory and the implications on practical design given in [BB03] and [KNM99], the following aspects of describing human work can be identified:

The central elements in Activity Theory are *activities*. An activity describes the relationship between a subject (an actor) and some kind of object (which are either tangible or intangible). The further description is cited from Bertelsen et al. [BB03]:

Activity is directed to satisfy a need through a material or ideal object. The subject's reflection of (including expectation to) this object is the motive of the activity. Human activity is carried out through actions, realising objective results. These actions are governed by the conscious goals of the subject. Goals reflect the objective results of action. Actions are conducted by series of operations, each 'triggered' by the conditions and structure of the action.

Activity Theory introduces a hierarchical relationship between *activity* (which is directed towards fulfilling a certain motive), *actions*, which are conducted by individual actors (subjects) to fulfil a certain goal (in the context of the activity) and *operations* as the most basic element. Operations describe the concrete way of execution an action and are triggered by specific conditions in the context of the respective action's goals. Instances of this hierarchy are not fixed. An operation in one context is considered an activity aiming at an explicit motive in another context. The distinction between an action and an operation depends on

the expertise and skills of the subject. An operation for one person is considered an action by another person, which needs further refinement into operations in order to break down complexity. An example is given in [RFW]:

The level of a particular activity depends less on the actual activity and more on the person or group undertaking it, eg, applying for a position may be

- *an activity for a first-time applicant and his family and friends*
- *an action for an experienced applicant familiar with the industry and the selection processes likely to be used*
- *an operation for a skilled user of online job sites with a well developed CV...*

These specifications allow the identification of the first elements for the conceptual model:

The WHAT aspect contains elements for describing what is done:

Activity The framing element for every modeling task. In general, a model contains exactly one activity. Activity thus can be considered synonymous to the notion of work process. Consequently, the activity is common for all participants involved in a work process.

Action Actions are the central building blocks of an activity. They are carried out by individuals, who want to reach a certain goal by accomplishing the action. Actions do only describe what has to be done and do not describe how it is performed.

The WHY aspect contains elements for describing the objectives of the WHAT aspect:

Motive The motive describes the objective of the whole activity. While it the motive ideally is the same for all involved participants, also individual differences may show up.

Goal Goals describe the objectives of actions. They may be different for each participant but should always contribute to the overall motive.

The HOW aspect contains elements for describing how certain actions are accomplished:

Operation Operations refine actions in terms of how an action is actually accomplished. Several sets of different operations are possible to accomplish a certain action. Operations are generic in the sense that they are unspecific to the framing activity and per se do not have a certain goal (but contribute to these of the actions they are contained in).

The WHEN aspect contains elements for describing the causal order and triggers of operations for a certain action:

Condition Conditions trigger operations in the context of a certain action and so form the bridge between the WHAT and the HOW aspect. Where applicable, they also determine the causal order of the operations to be accomplished. In the same manner, conditions form the bridge between activity and actions.

The WHO aspect describes who is involved in an activity (by performing certain actions):

Subject Any individual participant of an activity is a subject.

The USING WHAT aspect describes the objects involved in an activity:

Object An object is the element an activity is directed to. It has the potential to fulfil the needs of the involved subjects defined in the motive. Objects are either tangible (material) or intangible (immaterial) and are the most central ressource within an activity.

Stepping back to the theoretical foundations in Cultural Historical Activity Theory again allows to further refine and complete the conceptual model. Further aspects of human activity have been introduced by Engeström [Eng87] based on Leontiev's works (cf. Figure 4.2):

The triangle describes the structure of an activity and has to be considered to be orthogonal to the 'activity-action-operation'-hierarchy. While subject and object have already been described, the remainder of the triangle is made up of so-called *social mediators* relevant to the activity. These mediators are:

- *instruments*, which are either real, tangible means of manipulation for material or data or signs and representations of arbitrary kinds. The concept of instrument has to be seen in its broadest sense and *embraces both*

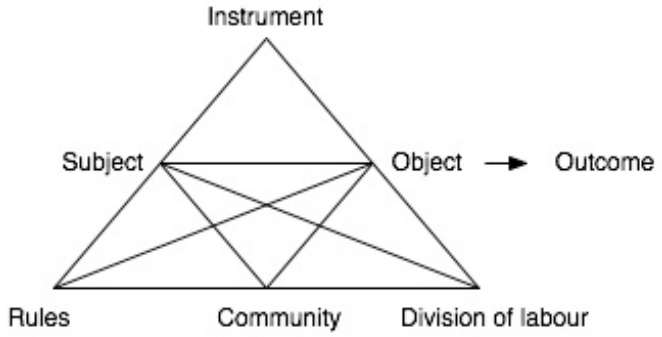

Figure 4.2: The mediational structure of an activity system (taken from [Eng87])

> *technical tools, which are intended to manipulate physical objects (e.g., a hammer), and psychological tools, which are used by human beings to influence other people or themselves (e.g., the multiplication table or a calendar).* [KNM99]

- *rules* which apply in the context of the activity
- *community* which forms the social context of the activity
- *division of labor* which refines the activity in terms of collaboration

Considering the mediational triangle and the checklists for practical application of activity theory given in [BB03] and [KNM99], the conceptual framework can be complemented with the following elements:

Refining the USING WHAT aspect Instruments introduce a greater variety of elements in the 'with what' aspect:

Tools are instruments, which are used within an operation to manipulate something

Material are instruments, which are manipulated within an operation. They are either tangible or intangible and can be preexisting, created and/or used/altered within an operation

Knowledge describes 'Know-What' that is needed to perform an operation (in contrast to 'know-how', which is modeled by skills).

Skills describe 'Know-How' necessary to perform an operation. In contrast to 'knowledge', a skill represents expertise necessary to perform something.

Refining the WHAT aspect The introduction of the 'division of labour' aspect enables refinement of the 'what' aspect in terms of distinction of own and others' actions in an individual model:

Producing Actions are actions which produce something, i.e. which have a definable, externalized output. This output is not necessarily tangible, materialized but may also be intangible like a computer file. The main distinctive property to an internal action is, that a producing action's results can be directly handed over to somebody else

Internal Actions are cognitive actions of an individual, with no visible effect on the outer world. An internal action's results are always internalized, within someone's brain and cannot be handed over directly to somebody else

Conversational Actions are actions which involve at least two subjects. Conversational actions are used when either material or information is transfered from one subject to another

Refining the WHO aspect the checklist of [BB03] claims for a description of *what different kinds of people are needed to produce [...].* Thus, the 'community' dimension can only be fully captured within the models by the introduction of additional elements:

Group of Subjects This element represents any formal or informal group of subjects. A formal group is, e.g. an organisational unit, a prototype for an informal group is, e.g. a community of practice [Wen99]

Role A role is defined by a set of necessary skills and can be taken by a subject who has these skills. Roles also may formally defined or may have developed informally. Formal roles are part of an organisation's structure, e.g. are linked to positions within an organisation. Informal roles can be found in every group of people and are never defined explicitly. The relevance of the latter category has been examined by [JRH05].

The NORMATIVE aspect To represent the 'rules', a way to express normative information within the models is needed:

Rules are used to define to normative aspects of an activity. They always operate on the elements used to model the activity and express issues that constrain the activity's execution. Rules are expressed in natural language in the course of modeling and may only contain references to elements that are contained in the respective model.

On a meta-level, the conceptual model considers the aspects of work processes given in Figure 4.3.

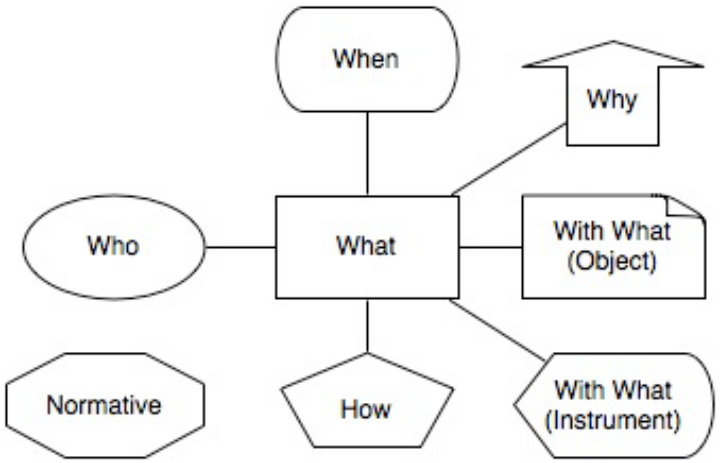

Figure 4.3: Conceptual model for procedural content - Meta-level

Consequently, as already indicated above, there are relationships between the elements of the conceptual model:

Hierarchical relationships always describe a 'part-of' relationship in the context of the model:

- an *activity* **consists of** a set of *actions*
- an *action* may **contribute** to several *activities*
- an *action* may **consist of** a set of sub*actions*
- an *action* **consists of** a set of *operations*

- an *operation* may be **carried out** in the context of different *actions*
- a *organizational units* **contains** one or more *subjects*
- a *subject* may **belong to** one or more *groups of subjects*

Semantic relationships describe the relationships mainly between but also inside the aspects of the model:

- an *activity* **is driven by** the *motive*
- the *object* **manipulated by** an *activity* (embedded into the corresponding *motive*)
- an *action* **is driven by** its *goals*
- *goals* may **support** or **contradict** other **goals**. This can be used to model subgoals of an action or dependencies between the goals of different actions
- *actions* and *operations* **are constrainted by** the *conditions* determined by the enclosing activity or action, respectively
- an *action* or an *operation* may be **performed** by either a *subject* or a *role* element
- a *role* **is determined** by the set of *skills* necessary to take it
- a *subject* **has** a certain set of *skills*
- a *role* **can be taken by** a *subject* (depending on the match of necessary and existing skills)
- an *action* or an *operation* may **need** a certain set of *skills* to be accomplished
- *rules* are **derived** from the *motive* of an activity
- *activities* are causally put into relationship using the **follows** connection
- *operations* are causally put into relationship using the **follows** connection
- an *operation* may need some **knowledge** or a **tool** to be performed
- *material* may be **required**, **altered** or **created** by an *operation* or an *action*

- *material* may be **composed of** other parts which are also represented as *material*

The conceptual model in detail consists of the elements and (organising) aspects given in Figure 4.3 and 4.4.

4.1.1 Check against other Reference Models

While Activity Theory allows the derivation of the conceptual model given above, the notions used (taken from activity theory) largely differ from those used in established reference models for work process design. Thus, the developed model has been checked against different reference models in this field (which basically spans the areas of process, workflow and task modeling) and try to map the elements given there to the elements of the conceptual model. Three reference models from different points of view on business information systems have been selected (see Figure 4.5):

- the WfMC Reference Model for workflow modeling [Wor95] (covering the IT perspective)
- the Task Modeling Meta Model of van Welie for task modeling [vW01] (covering the human perspective)
- the Process Modeling Comparison Framework of Soderstrom et al. for process modeling [SAJ+02] (covering the organisational, economic perspective)

The meta-model level of the conceptual model is used to classify the elements of the selected reference models. Each reference model is then compared to the conceptual model on element level for each aspect. Consequently, the elements of the selected reference models are considered to reside on model level in the context of this comparison - even when they are abstracted from actual modeling approaches and originally are designed for explanatory purposes only. This reinterpretation is valid, as comparison is performed on conceptual level and not by using actual instances of processes. The selected reference models abstract from actual approaches for modeling. Thus, they are better suited for a comprehensive check of the conceptual models's coverage of necessary elements.

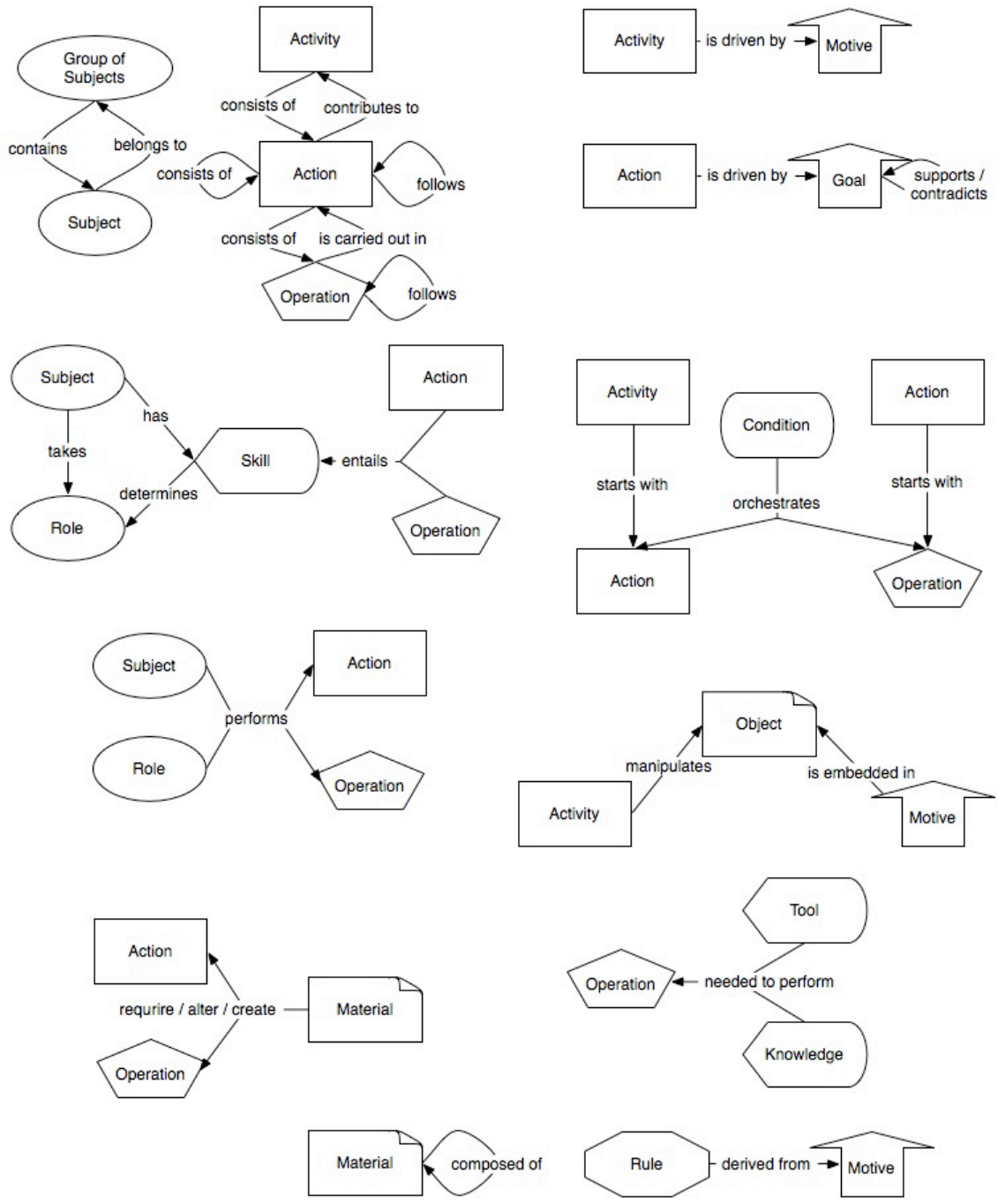

Figure 4.4: Conceptual framework for procedural content - A detailed view on model elements

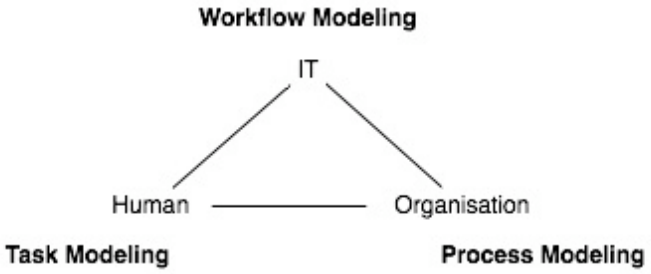

Figure 4.5: Modeling approaches for relevant aspects of business information systems

The WfMC Reference Model

The Workflow Management Coalition has developed the WfMC Reference Model [Wor95] as a framework for workflow systems. It is designed to support the development of workflow management systems by specifying concepts, terminology and general structure. Focusing on concepts, workflows represent the execution aspect of an process (often also IT-centered, considering the user to be a means for data-input and manipulation). The WfMC Reference Model (see figure 4.6) is a widely accepted frame of reference for workflow modeling languages and thus is considered in this context.

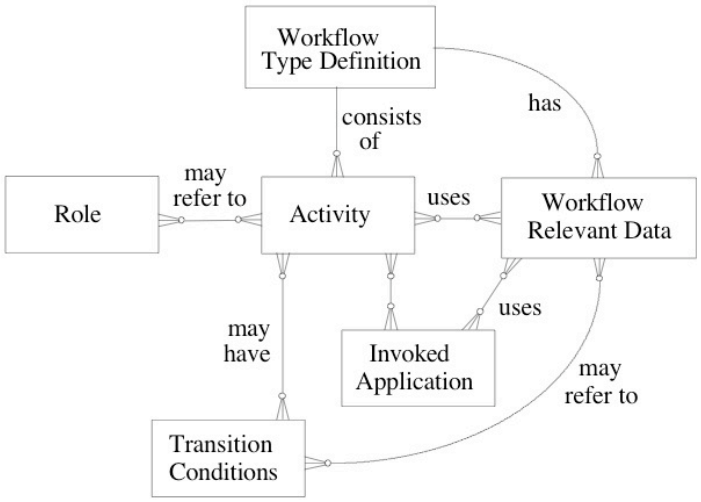

Figure 4.6: WFMC Basic Process Definition Meta-model (taken from [Wor95])

The concepts of the WfMC Reference Model are mapped to those of the conceptual model in table 4.1.

Table 4.1: Mapping of the WfMC Reference Model to the Conceptual Model

WfMC Reference Model	*Conceptual Model*
Workflow Type Definition	Activity
Activity	Action (without considering the WHY-Aspect), Operation
Role	Role
Transition Conditions	Conditions
Workflow Relevant Data	Material (restricted to materials containing data)
Invoked Application	Tool (restricted to computer applications)

Models based on the WMfC reference model can be fully represented using the conceptual framework proposed in this work. While some elements can be mapped directly (like roles or conditions), others require additional restriction of their counterpart in the conceptual model (like tools, which have to be restricted to computer applications, as the WfMC model only considers those).

However, a comprehensive mapping is not possible the other way round. The WfMC reference model especially lacks possibilities to represent the WHY-aspect (which is hardly relevant for pure execution of a workflow). Other aspects are not fully covered in detail. The USING WHAT aspect lacks knowledge and skills and thus elements used to represent the human facet. This is rooted in the IT-background of the WfMC-model. The idea of abstracting processes from executing persons has impact on the elements of the WHO aspect, where the WfMC reference model does not provide any means to represent persons but only enables specification of abstract roles. It also does not provide any means to model the NORMATIVE aspect.

Recapitulating, the WfMC reference model does not uncover any shortcomings of the conceptual model proposed in this work. Models based on the WfMC reference model can be fully represented in the conceptual model.

Task Modeling Meta Model

The reference model of van Welie [vW01] (see figure 4.7) is a representative for the wide field of task modeling languages (see figure 4.7). Task modeling languages historically focus on the boundary between human action and IT support (from a user's point of view) und thus are often used for the design of interactive systems.

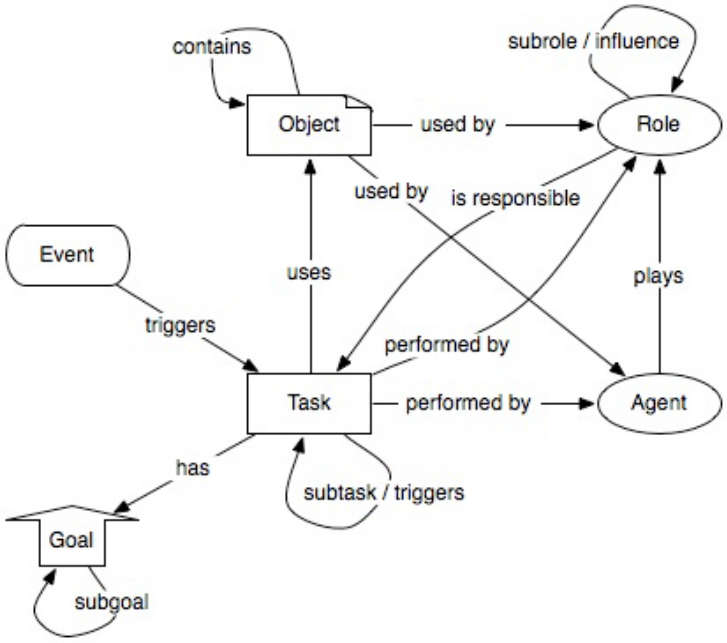

Figure 4.7: Meta Model for Task Modeling (adapted from [vW01])

The elements of the task modeling meta model are mapped to those of the conceptual model as follows. The mapping of relationships is described in table 4.2.

Again, the elements of this reference model can be fully mapped to the conceptual model. However, it lacks the layer of concrete operations (i.e. it only specifies what to do, but not how to do it). For task modeling, this is of no interest, because tasks and operations are often not confined or distinguished in this area (e.g. very detailed specifications of tasks on operational level are given in [MPS02]).

The associations defined in van Welie's approach can be mapped to those of the conceptual framework directly in most cases. However, there are some

Table 4.2: Mapping of the Task Modeling Meta Model to the Conceptual Model

Task Modeling Meta Model	*Conceptual Model*
Task	Action
Event	Condition
Goal	Goal
Goal	Goal
Object	Material, Tool
Agent	Subject
Role	Role

definitions that require special attention:

subgoal can be represented by the supports/contradicts relationship. While on the one hand this is more fine grain, the explicit information of being a subgoal is lost and can only be reconstructed implicitly in the context of the action's other goals or the goals of possible subactions.

is responsible (between task and role) is the reverse relationship of 'performs'. It is not available in the conceptual model because relationships with a cardinality other then m:n are implicitly reflexive in the conceptual model (in contrast to m:n relationships like between actions and operations, where both directions are modeled separately)

subrole cannot be represented in the conceptual model as it could not have been justified against activity theory

influence (between roles) influencing always involves a conversational action and has to be represented using this element for reasons of consistency

used by (between object and role) can only be represented by a mediating operation, as using something always includes someone who 'uses'. For reasons of consistency, an operation as part of a producing action as to be used.

The only shortcoming of the conceptual model against the task modeling meta model of van Welie is the lack of hierarchical relationships between roles.

Wether this kind of relationship is really necessary will be evaluated in practical evaluation scenarios. For now, it has not been considered, because no justification in activity theory or its applications has been found (although it seems not to contradict activity theory per se).

Recapitulating, the task modeling meta model matches the conceptual model to a large extent (presumably because they both have their foundations in a human-centred view of work processes). However, the task modeling meta model lacks the HOW aspect of work models and thus does not allow explicit distinction between the WHAT and HOW aspects. It also does not provide any means to model the NORMATIVE aspect.

Process Modeling Comparison Framework

The process modeling comparison framework developed by Soderstrom et al. [SAJ+02] covers the research area of process modeling (see Figure 4.8) in this comparison. No existing modeling language actually provides such a comprehensive feature set as the comparison framework. It seems to be appropriate for reflection on the conceptual model, because it is based on and has been checked against the established modelling languages *Event-driven Process Chains*, *UML State Diagrams* and *Business Modeling Language* (similar to SDL) - references to these approaches are given in [SAJ+02]). Each of these languages represents a different approach to process modeling (activity-oriented, state-oriented and communication-oriented). By unifying them, the comparison framework creates a comprehensive picture of this research area.

Approaches in process modeling in most cases take an organisational, economic perspective when modeling work. The elements of the conceptual frameworks also cover this perspective to a large amount (see table 4.3).

The process modeling comparison framework provides more modeling elements than the conceptual model. However, the expressiveness of the comparison framework is significantly higher only in one aspect. The conceptual model does not cover the WHERE aspect, while the process modeling comparison framework offers the 'location' element to express where an event takes place.

The process modeling comparison framework just like the conceptual model explicitly specifies aspects of modeling (like WHO, HOW, etc.). Some of the elements are assigned to different aspects in the two models and the aspects them-

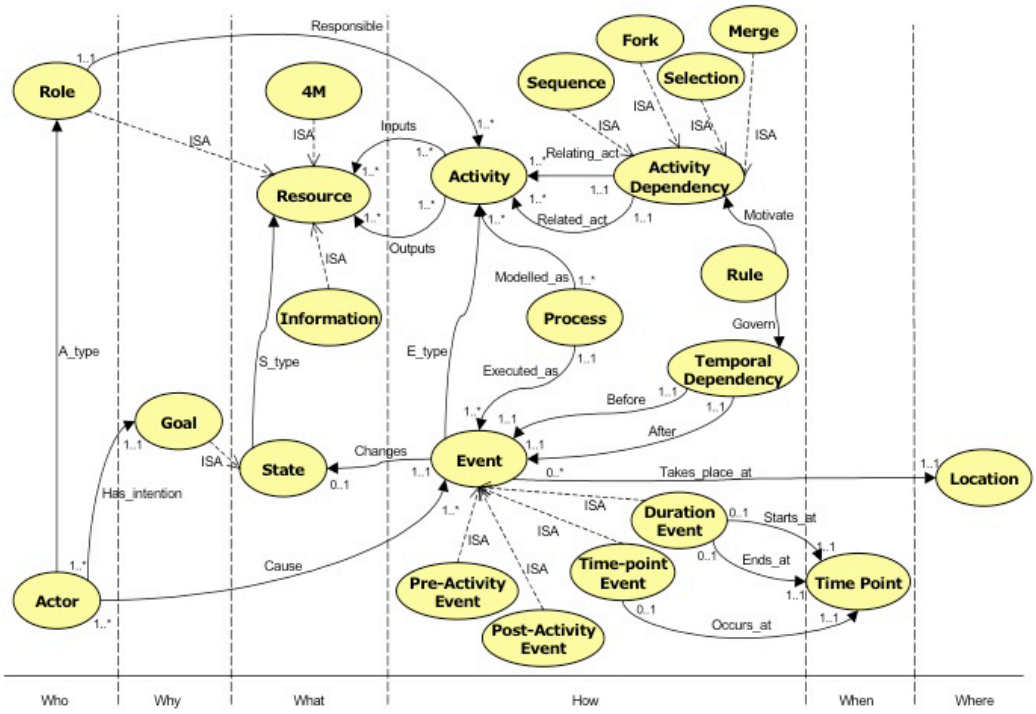

Figure 4.8: Meta Model for Process Modeling (taken from [SAJ+02])

Table 4.3: Mapping of the Process Modeling Comparison Framework to the Conceptual Model

Process Modeling Comparison Framework	*Conceptual Model*
Role	Role
Actor	Subject
Goal	Goal
State	-
Information	Material (only intangible)
Resource	Material
4M	Material (unknown subtype)
Activity	Action
Process	Activity
Event	Condition
Activity Dependency	Condition
Rule	Rule
Temporal Dependency	Association 'follows'
Time Point	-
Location	-

selves are partially interpreted differently. This is especially true for the WHAT and HOW aspect. All elements representing something people do or perform are included in the HOW aspect in the comparison framework. The WHAT aspect in the comparison framework corresponds to the USING WHAT aspect in the conceptual model. While the respective elements basically match completely, the comparison framework does not explicitly distinguish between the conceptual model's WHAT and HOW aspects. The WHEN aspect is also interpreted differently. In the conceptual model, WHEN aims at representing causality and temporal relationships. In the comparison framework, causal and temporal relationships are included in the HOW aspect. The WHEN aspect there is used to define some point in time (to model special kinds of temporal events). This capability is not included in the the conceptual model. Temporal information is specified in rules there when necessary (avoiding redundancy).

The process modeling comparison framework is lacking elements to represent operations and tools. This is not surprising, as these elements are relevant for execution only, which is in the scope of workflow modeling and rarely is considered in the more abstract process modeling domain.

Recapitulating, the process modeling comparison framework offers a comprehensive set of elements when compared to the conceptual model. While the comparison framework even goes beyond the expressiveness of the conceptual model in some aspects, it lacks the capability to represent classic workflow elements like operations and tools.

4.1.2 Summary

In this section, a conceptual model for work processes has been developed based on Activity Theory. The model has been checked against established reference models to identify semantical equivalences between the elements of the different approaches. Besides some reality check, this results in more extensively defined elements in the conceptual model (by linking to existing concepts in other approaches).

Reviewing work process modeling in information systems research, Green and Rosemann [GR00] derive the following aspects a model of a work process has to contain:

- A representation of control flow

- Information about the organisational units that are involved in the process
- The objectives and results (output) of the process
- Information about resources and knowledge used in the process
- References to descriptions of existing functions within an enterprise

These aspects can be summarized in the question *'who does what how and when, using which ressources and why?'*. They also correspond to the aspects defined in the conceptual model (except 'references to descriptions of existing functions', which can be realized by attaching additional, natural language information to the 'group of subjects'-element). There is no explicit demand to be able to model rules. However, rules can be considered to be implicitly contained in the other aspects, as rules might affect them. Green and Rosemann do not specify a reference framework but only give guiding suggestions on how to model work and do not claim formal completeness of their aspects.

In contrast to established approaches, the model presented here does not determine the set of elements the modeler has to use. The elements defined in the conceptual model may serve as a basic set of constructs to be used for modeling (their designation, however, has to be adapted to the modeler's vocabulary). In addition, it allows the introduction of arbitrary elements to explain the individual view onto a work process. Modelers are not forced to adapt and map their view of their working environment to a different one for external representation. Having said this, it is obvious, that the resulting model will not be deterministic [FHL+98], i.e. will allow to build semantically equivalent model instances - different instances that represent the same phenomenon of reality. The conceptual model in this context is used to represent the user-defined elements in order to get to a common foundational model.

4.2 Content

The contents of this section are based upon the works of Auinger and Stary on content engineering for didactically motivated knowledge transfer ([AS05b], [AS05a]). The concepts discussed here have been implemented in the eLearning-platform *Scholion*, which will be complemented with this work.

Following Auinger and Stary's approach, learning content is divided into small content *blocks*. The division criterion is the didactic intention of a content part (which can be assembled of text, images and other multimedia elements). The following block types have been considered relevant in eLearning settings [AS05b]:

- Motivation
- Definition
- Example
- Case Study
- Code
- Exercise
- Reference
- Theorem
- Overview
- Directive
- Supplement
- Test
- Interaction
- Summary
- Generic Content

Every content block is assigned a specific block type augmenting it with additional semantics. Several applications, from consistent didactically motivated authoring of content to filtering content while learning, can be built upon this concepts.

Blocks are assembled to *learning units*. Learning units are self-contained regarding content and didactics. They present one topic, where the granularity of topics should be equivalent to a presentation of content in 15 to 45 minutes.

Courses are an assembly of learning units to produce a comprehensive presentation of a certain topic area, where granularity is equivalent to that of course in traditional, lecture-based settings.

On block level, the presented concept introduces three *levels of detail*. Every block can be specified on different levels of detail (LOD). These LODs facilitate different learning depths:

LOD 1 presents content in keywords (like on slides)

LOD 2 presents content in full text (like in lecture scripts)

LOD 3 provides additional (background) information or further reading to the respective content

In terms of individualisation, Scholion provides users with the concept of *views*. Views can be interpreted as an individual overlay of the learning content, on which learners can mark parts, annotate and add links to external or internal ressources.

Figure 4.9 shows the relationships between learning units, blocks and levels of detail. A single block may be used in several learning units. Blocks may also be nested in each other within a learning unit, so that hierarchical structures can be created.

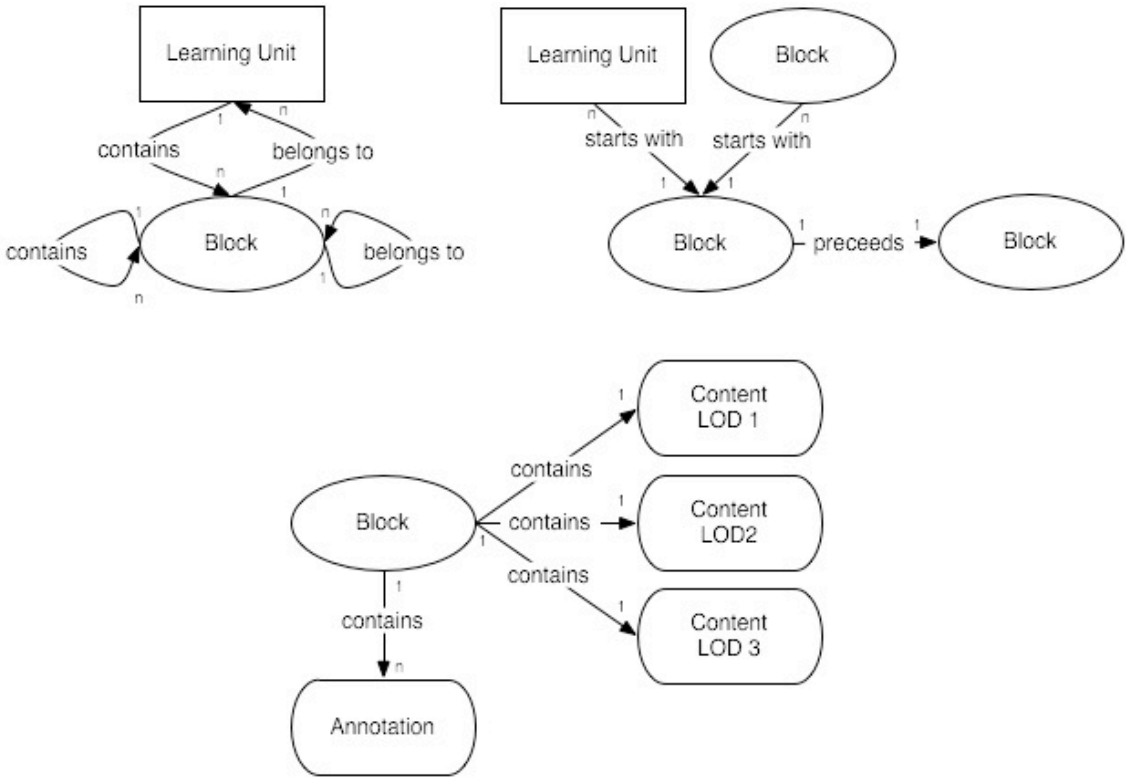

Figure 4.9: Conceptual Content Framework

4.3 Communication

Support for communication is also based upon the concepts applied by Auinger and Stary ([AS05b], [AS05a]) in the development of the eLearning-platform

Scholion. They identify three basic types of communication in an eLearning-setting:

asynchronous, unidirectional implemented in the *infoboard*, which is structured like a bulletin board, on which teachers publish information for students

asynchronous, bidirectional implemented using a *forum* for discussion among both, students and teacher

synchronous, bidirectional implemented as a *chat*, which allows real time communication among students and teachers.

Infoboard, forums and chats are attached to a course (as defined in the content section). Forums are structured using *topics*, on which *discussions* can be opened. Chats contain *chatrooms* for structuring along discussion topics. Discussion entries and chat entries are ordered temporally. The Infoboard (and its entries), chatrooms, discussions and discussion topics are not ordered.

Figure 4.10 shows the communication model structure used in this work. Discussion entries can be nested to realize a hierarchical structure of questions and answers in a discussion.

4.4 Meta-Data

The following structural meta-data is stored with every element of the models given above:

- Author / Owner
- Creation Date
- Date of last change

4.5 Inter-model Aspects

Until now, the three areas of content (process, content, communication) have been presented and analysed separately. These three areas have to be interconnected semantically to facilitate OL.

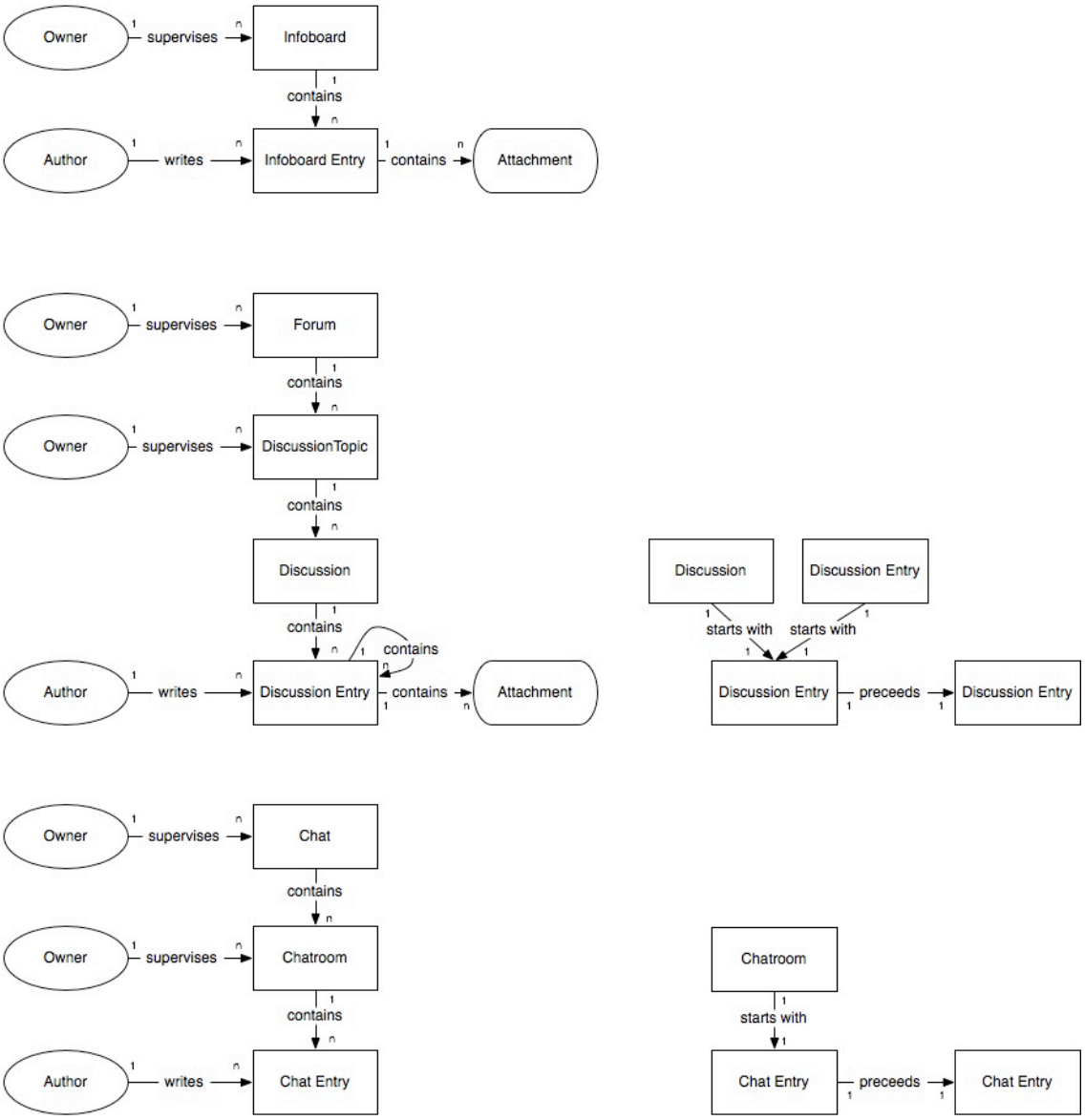

Figure 4.10: Conceptual Communication Framework

The concept of intertwining content and communication has already been applied in the development of Scholion. For example, it is possible to link parts of the content to discussions or chats (and vice versa).

In this work, the process dimension is intertwined with content and communication. Thus, it is possible to:

- link specification or background material to work process models
- discuss about work processes (synchronously and asynchronously)
- put content into the context of work processes, in which it is considered relevant

Besides that, the approach allows individual specification of arbitrary relationships (with freely definable semantics). Users in this way can express individually perceived relations by establishing links between elements (of process, learning and communication).

4.5.1 Extensibility

Extensibility is relevant to this work in two cases. The first - and more generic - case is extension by *additional* content types (like the already defined *process*, *content* and *communication*). This is necessary whenever the OL platform is extended with new features that require the incorporation of a new information aspect.

The second extension point is *inside* already existing content types. This is on the one hand relevant for communication, where new communication channels can be established with this means. On the other hand - and more important for OL - domain specific extension of the models in content and process area is also possible. By this means, users are enabled to introduce (and define) concepts of their organisational domain or their very own mental model. As already stated in the first part of this work, this is considered a very relevant feature to bridge the gap between individual and organisational learning. In this area, recent work on formalisation of flexibility in (work) process modeling is presented in [RR06] (conceptually) and in [MRRvdA06] (applied to C-EPCs).

Chapter 4: Summary

In this chapter, the content types relevant for OL have been put into mutual context with the requirements on content representation in Scholion. The resulting models allow to represent content of the specified types in a structured, block-oriented form. Decomposition of content follows different criteria for each content type. The resulting elements of content are either semantically motivated (for learning content), purely hierarchical and causal (for discussion content) or both (for description of behaviour in procedural content). Considering these results, a concept of data representation is needed, which allows to express content of arbitrary type and structure in a consistent form.

Chapter 5

Data Representation Concepts

Now that we have all this useful information, it would be nice to be able to do something with it. (Actually, it can be emotionally fulfilling just to get information. This is usually only true, however, if you have the social life of a kumquat.)

UNIX Programmer's Manual

Objectives of Chapter 5

The goal of this chapter is to select and describe a suitable concept for representation of content. The selection is triggered by the requirements identified in part I. A detailed description of the concept is necessary to allow mapping of content elements to actual representation

The concept for content representation is developed in the following based upon the requirements identified in the first part. Recapitulating, the requirements of OL research on learning content representation are:

- representation of different content types
- storing a single content element in different, arbitrary forms of representation
- adding individual information and remarks to content

- putting content elements into arbitrary relationship with each other
- attaching structural and domain-specific meta-data to content elements

In addition, the concepts of Scholion required to find a form of representation that provides:

- arbitrarily definable blocks of content
- arbitrarily definable associations of blocks of any type
- arbitrarily definable forms of representation of content blocks

Brought down to a more technical view on data representation, this means that the selected concept:

1. must not rely upon fixed data models but has to allow description of these as part of the representation
2. must allow to build semantically linked structures of elements (in other terms: conceptual graphs [Sow84] [Sow00])
3. must allow arbitrary representation of the actual content and even several distinct and/or complementing representations for the same element

The only concept, that meets all three requirements in a consistent form is that of *Topic Maps* [ISO06a]. Topic Maps are a means to represent semantic networks (meeting requirement 2), where actual semantics of both content elements and relationships are arbitrary but explicitly definable within the topic map (meeting requirement 1). Topic Maps also allow to attach arbitrary (even multiple) representations of content elements again with arbitrary but definable semantics (meeting requirement 3).

A second candidate for data representation is using a combination of the semantic web standards *RDF and OWL* [W3C06]. With these technologies, the three requirements given above can also be met. RDF/OWL however follows a distributed approach for expressing semantics by attaching semantic meta-data to the actual content representation. In contrast to topic maps, no explicit representation of the content structure is available but has to be derived from content-meta data. Topic Maps - following a centralized approach to expression of semantics - better suit this work, where focus is on structure of content.

RDF/OWL explicitly focus on representing ontologies, an area which is not covered by the topic map standard. However, topic maps can be used to represent ontologies because of their generic approach to data representation. In order to do so, topic map constructs for building ontologies have to defined by the system designer - topic maps do not provide any themselves. The constructs used for ontology description are described in section 8.1. They have been inspired by those introduced by RDF/OWL.

Both approaches are briefly presented in the following. However, focus is on topic maps, because they will be used for data representation. RDF and OWL are reviewed in terms of complementing or conflicting concepts - especially for representation of ontologies. A brief overview about recent research in the area of intertwining Topic Maps and RDF/OWL points out possible approaches for complementary usage.

5.1 Topic Maps

Topic Maps are a standardized means to represent semantic networks and associating the contained concepts with information ressources. Their expressive power corresponds to those of conceptual graphs and goes beyond by incorporating the expressiveness of indices. The index concept enables to link concepts to instances, i.e. representations of corresponding information [Pep00].

Topic Maps per definition are 'ontology-agnostic', i.e. they are capable of representing any kind of information in any context (or: to express *anything about anything whatsoever* [Vat04]). Therefore, they are an ideal means of representation in this work, where it is necessary to express content structures (instances), models, meta-models and also links to the actual information representations.

Topic Maps and the corresponding XML representation format XTM [Top01] have been standardized by ISO [ISO02] in 2002. This work is based on the revised version of the standard (v2.0), which has been approved in the beginning of 2007. The new version of the Topic Map Standard consists of multiple parts, of which the most relevant are the Topic Map Data Model (TMDM) [ISO06a], a formal meta model for topic maps, and the corresponding XML representation format XTM 2.0 [ISO06b].

Topic Maps consist of several building blocks, which can be used to model

concepts, the relationships among them and the links to actual instances of concepts. These building blocks are reviewed in annex B, including a reflection of how they will be used in the context of this work. A topic map basically consists of topics, associations (see figure 5.1) and occurrences (as the bridge to the 'outer world'), which are intertwined as described in annex B (also see figure 5.2. Regarding terminology, topics correspond to concepts, associations represent relationships and occurrences map to instances.

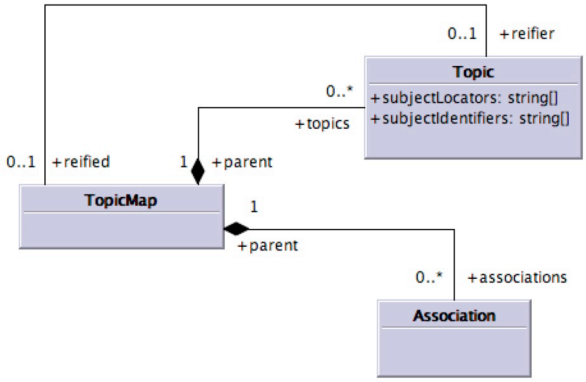

Figure 5.1: Topic Maps - Basic Elements (taken from [ISO06a])

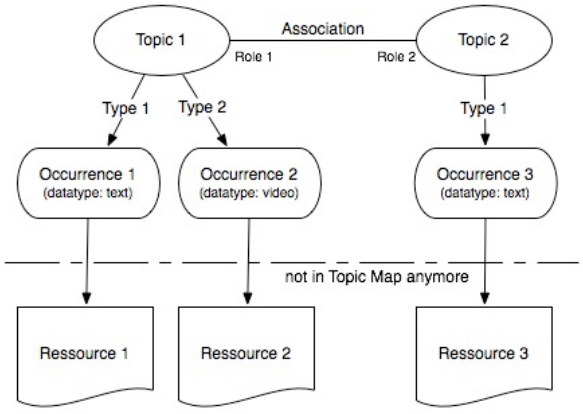

Figure 5.2: Topic Maps - Structural Overview

5.2 RDF & OWL - Competitior or Complement

Another approach to semantic augmentation of information ressources is RDF/OWL. RDF and OWL have been standardized by the W3C [W3C06] in the course of the Semantic Web Initiative and aim at fulfilling similar purposes as topic maps with a rather different approach.

5.2.1 RDF & OWL Overview

RDF s short for for *Ressource Description Framework* and has been designed as language *for representing information about resources in the World Wide Web. It is particularly intended for representing metadata about Web resources [...]. However, by generalizing the concept of a 'Web resource', RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web* [W3C04b].

In contrast to topic maps, RDF basically starts bottom up at the occurrences (here: web resources) and describes (a) the semantic type (basically similar to topic types) of a resource (e.g. an resource describing a 'Person') and (b) meta information (basically similar to topics and associations) of a resource (e.g. the person described in the resource is 'Mr. X' and can be reached under 'mrx@foo.com'). However, RDF lacks the capabilities necessary to specify an ontology (in terms of concepts and associations). A first step towards ontology specification is RDF-S (RDF Schema) which *is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes* [W3C04a].

However, to provide full ontology support, OWL has been introduced. OWL is an acronym for *Web Ontology Language* and is builds on top of RDF to provide full ontology support. *OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. OWL has more facilities for expressing meaning and semantics than XML, RDF, and RDF-S, and thus OWL goes beyond these languages in its ability to represent machine interpretable content on the Web* [W3C04a]. OWL provides full support to model and formally represent ontologies, that is, modeling concepts, associations and explicit formulation of rules and constraints for usage of ontology concepts (which is not explicitly possible when using topic maps).

5.2.2 Towards integration with Topic Maps

The first approach to represent Topic Maps using RDF constructs has been made by Ontopia [Ont03] in 2003 but has been abandoned shortly afterwards as RDF had proven to provide too little expressiveness to fully represent Topic Maps. This lack of expressiveness has been overcome by the introduction of OWL.

In [Vat04], Vatant has shown how OWL can be used to replicate the topic map constructs and represent topic maps. He claims this to enable the introduction of ontologies into the topic map world (which - as mentioned above - are considered 'ontology-agnostic'). He also argues for compliance with the established and widely adopted OWL-standard for modeling ontologies:

In fact topic maps would indeed gain effectiveness and interoperability either through explicit formalization of ontologies specifically built and dedicated for topic map control, or through declaration of commitment to predefined ontologies, not specifically designed for that use. In either case, using OWL ontologies should be considered as the most interesting choice.[Vat04]

More recently, Cregan [Cre05] has proposed a comprehensive model of topic map representation in OWL. Using OWL for building ontologies also overcomes the described drawback of being not able to define constraints on topic types (e.g. usage of a type solely with a specific association role or an occurrence). OWL includes the concepts of *restrictions*, and so enables constraining association roles and occurrences in the intended way.

Topic Map instances are defined in a meta model specified in OWL. The topics (actually only those topics which are directly associated with at least one occurrence) and associations among them are represented using RDF (based upon the OWL ontology). However, this mapping puts down the flexibility of topic maps. It is not possible any more to recursively step up and down modeling levels to extend and refine the (meta) model. This problem is also adressed in [Cre05] as the major drawback of using OWL for topic map representation - Cregan suggests a possible solution which cannot be automatically processed using OWL engines (in contrast to a pure topic map solution). Nevertheless, it seems to be possible to translate a stable topic map (including its meta model) into an RDF/OWL-assembly, being able to provide compliance to the most widespread standard for semantic augmentation on the web (while the same is true the other way round, RDF/OWL-constructs can also be represented in topic maps, as mentioned in [Rat03] and implemented in the Ontopia Omnigator [Ont06]).

An comprehensive overview and comparison of topic maps and OWL is also available in [Raf05].

Chapter 5: Summary

Topic Maps have been identified as a suitable means for content representation based upon the requirements identified in part I. Topic Maps are open to every form and structure of content. They provide means to explicitly define content models including available elements, associations and forms of content representation. They are an ideal candidate to realize flexibility in content representation for OL. For actual use in this work, models for the specified content types are needed. The topic map concepts will be used to represent these models. This is subject of the following part.

Part III

System Design & Implementation

Chapter 6

System Design Overview

The primary purpose of the DATA statement is to give names to constants; instead of referring to pi as 3.141592653589793 at every appearance, the variable PI can be given that value with a DATA statement and used instead of the longer form of the constant. This also simplifies modifying the program, should the value of pi change.

FORTRAN manual for Xerox Computers

Objectives of Chapter 6

The main goal of this chapter is to create a high level software design based on the results described in the former parts. In addition, the technical constraints on implementation for Scholion-integration are identified. The results given here provide the context for the detailed design and implementation described in the following chapters.

Figure 6.1 gives an overview of the software system architecture. The central part of the system is the topic map engine (cf. chapter 7). It has been implemented based upon the specification of ISO Topic Maps as described in annex B. An interface has been introduces for flexibility of persistent topic map storage. This interface has been implemented for XTM and relational databases via Hibernate (cf. section 7.1).

On top of the generic topic map engine, an *OL layer* has been implemented (cf. chapter 8). It contains the models defined in chapter 4 and provides access-

routines for easier management of OL content (making the topic map representation invisible for users).

The peripheral components, the *Scholion Data Importer* (cf. section 8.3) and the *Integration into the Scholion platform* have been designed conceptually so far and are only implemented prototypically in the course of this work.

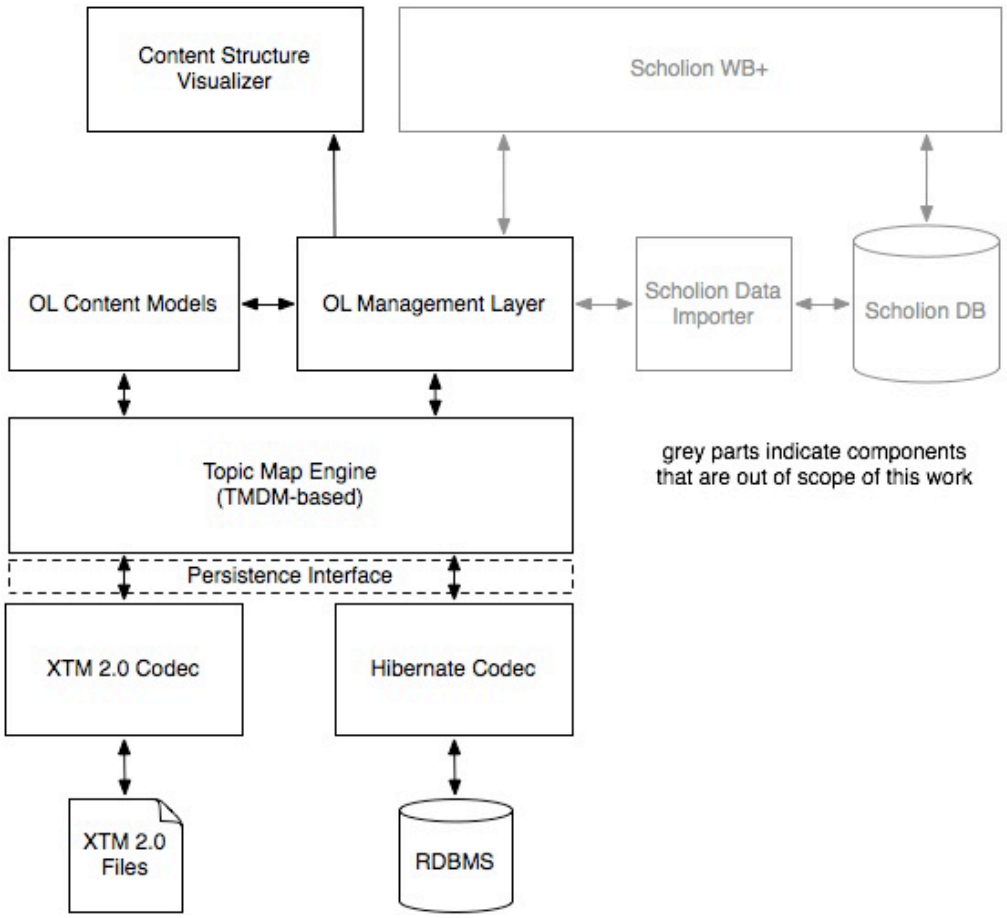

Figure 6.1: Architectural Overview

6.1 Technical Constraints for Scholion-Integration

Some fundamental design and implementation decisions have to be made before going into detailed system design. The existing IT infrastructure within and around the Scholion plattform sets some constraints on system design:

Java The use of Java as the language for implementation is a requirement on all software systems integrated with Scholion. Consistent usage of programming languages makes maintenance easier and allows extension and debugging of the software by dedicated project members or students.

Encapsulated Data Representation While the work presented here aims at substituting the current data management layer of Scholion, for now it is considered an extension, that provides additional features. As this work works with rather low-level data manipulation routines, it is crucial that - in the prototyping stage - data representation between the software system of this work and Scholion is strictly encapsulated. Furthermore, data encapsulation is also a general requirement on new extensions developed for Scholion.

Hibernate Integration Scholion's data representation is based upon a RDBMS (relational database management system). Hibernate [Red07] is used for all extensions to map Java objects to data base entries. In this way, a consistent, yet flexible data management layer is provided.

Transfer of existing Scholion content The capability to import existing content (both, learning content and communication data) from the already available data sources of Scholion is necessary to allow a smooth transition to the new, topic map based data management layer.

XML Data Export Scholion content can be rendered to XML (based on standard learning content formats) to provide a means for easy content exchange between instances of Scholion or even other platforms. As Topic Maps provide a standardized format for XML representation (XTM), this format is used to export content to XML.

Chapter 6: Summary

This chapter gives an overview of the software system's design. It sets the frame of reference for the software components presented in the following chapters. Besides that, the technical requirements for integration into the Scholion Platform are specified. These requirements directly affect system design and have to be considered during implementation of the components.

Chapter 7

Topic Map Engine

A man from a primitive culture who sees an automobile might guess that it was powered by the wind or by an antelope hidden under the car, but when he opens up the hood and sees the engine he immediately realizes that it was designed.

Michael J. Behe in 'Molecular Machines: Experimental Support for the Design Inference'

Objectives of Chapter 7

In this chapter the implementation of the topic map data model (presented in annex B) is described and complemented with routines to create, manage and access a topic map. The engine allows to map the content models defined in the former part onto topic maps. Besides that, approaches to store and retrieve topic maps to and from different data formats are developed. The latter is necessary for integration into the Scholion data management concept.

The Topic Map Engine implemented in this work complies to the ISO Topic Map Data Model [ISO06a]. Figure 7.1 shows the classes, which have been created to represent the elements of a topic map. This structure exactly maps to that given in the topic map data model (cf. annex B), except the introduction of an explicit representation of *scopes*. The *Scope*-class has been introduced for more convenient management of scopes. It allows to not only to retrieve the topics that constitute the scope but in addition the statements which are contained

in the scope (which would otherwise only be possible by iterating through all statements).

Figure 7.1 also shows the *Manager*-class and the *Utils*-class. The *Utils*-class only contains routines for internal management of the engine. The *Manager*-class provides a unified interface to the topic map engine. Although all topic-map-constructs can also be created manually, using the *Manager*-class assures consistency of the topic map. *Manager* keeps track of topics used as types (for other topics, associations, occurrences, etc.) and scopes. It also allows more efficient management of a topic map by providing routines for convenient topic map construct manipulation. Using the *Manager*-class, a topic map can be created more easily and understandable (for details of usage see annex C). Every topic map is managed by an instance of the *Manager*-class, which allows to conveniently administer several topic maps.

Figure 7.2 shows the classes of the topic map engine in more detail. The attributes and associations given there show that the implemented structure is equivalent to that of the topic map data model given in section B (both figures are generated directly from source code).

The topic map engine in a means of topic map representation in Java objects. When working with topic maps, it is crucial to provide means of persisting and retrieving them from storage. A basic design requirement was to allow for arbitrary persistence technology to allow attaching the system to various data sources and sinks. The implementation of the persistency interface and two of its implementations are described in the following section.

7.1 Persistency

Topic map persistency using arbitrary technology is a necessary requirement for interoperability. An interface has been introduced to remain flexible. It has to be implemented for each persistency technology. The interface provides methods for both import and export of topic maps. In the course of this work, two codecs using the interface have been implemented in student projects:

XTM 2.0 XTM 2.0 is a standardized XML dialect [ISO06b] for storage of topic maps. As XTM is the only standardized means of topic map data exchange, this codec is necessary to enable usage of the engine's representations with third party applications, editors and/or visualizers.

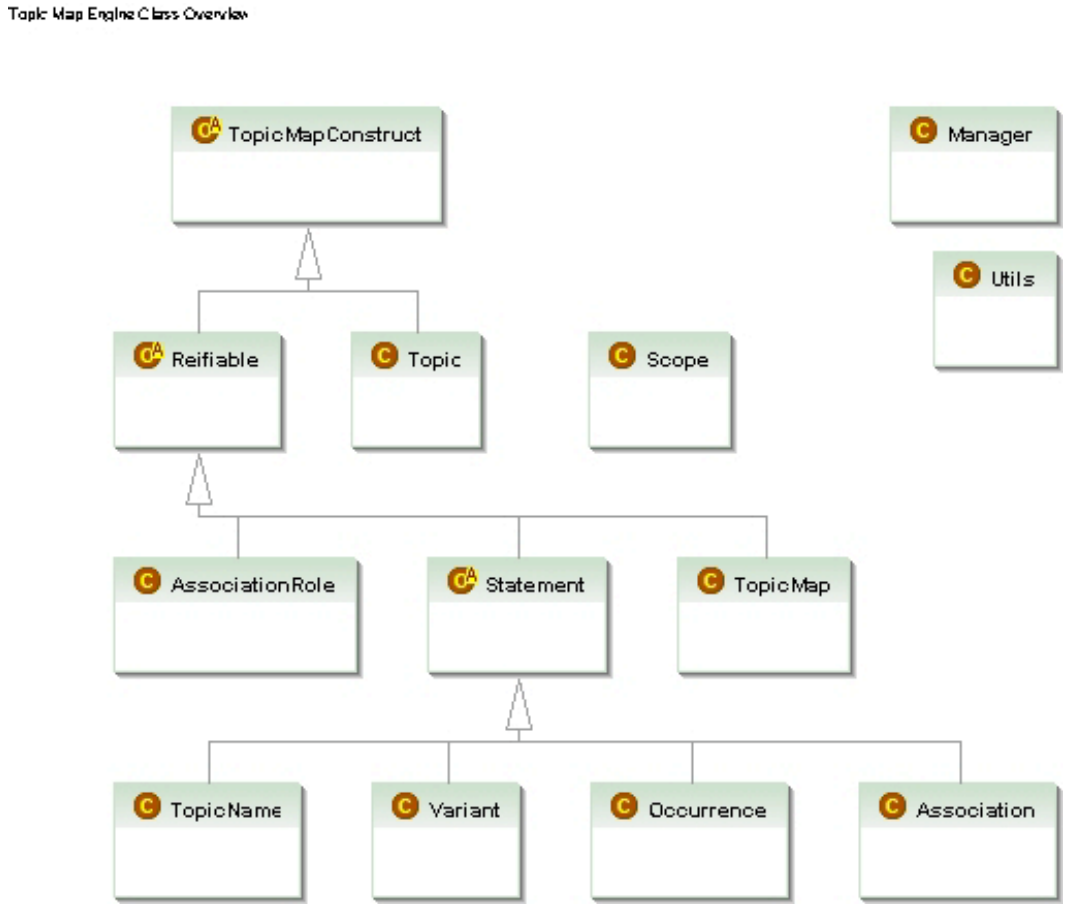

Figure 7.1: Topic Map Engine - Class Hierarchy Overview

Hibernate Hibernate [Red07] is an open-source framework for mapping of (Java-)objects to relational databases. Hibernate is already used in Scholion development and therefore has been chosen as a means of topic map persistency. Using Hibernate to store and retrieve topic maps enables seamless integration into Scholion's data management logic and allows to use an arbitrary RDBMS for topic map persistency.

Both implementations store topic maps represented within the engine and load them from storage (with subsequent recreation of the internal structure the engine uses for management). The implementations have been tested and work as expected (i.e. they produce valid XTM files and Schema-compliant data base entries, which contain every information represented in the original topic map).

The persistency interface is also used to output topic maps graphically using the *GraphViz* graph layouting toolset [GN00]. An encoder to GraphViz's dot-format has been developed for straight-forward and flexible visualisation of topic map contents. This component has been used for implementation of the visualization application described in chapter 9. A decoder from dot-format to topic maps can not be implemented because the dot-format does not represent all information necessary to reconstruct the whole topic map.

Chapter 7: Summary

In this chapter, the fundamental software component of this work - the topic map engine - has been described. The topic-map-engine is the foundation for all software components described in the following chapters. It especially allows to technically represent the OL content models already presented.

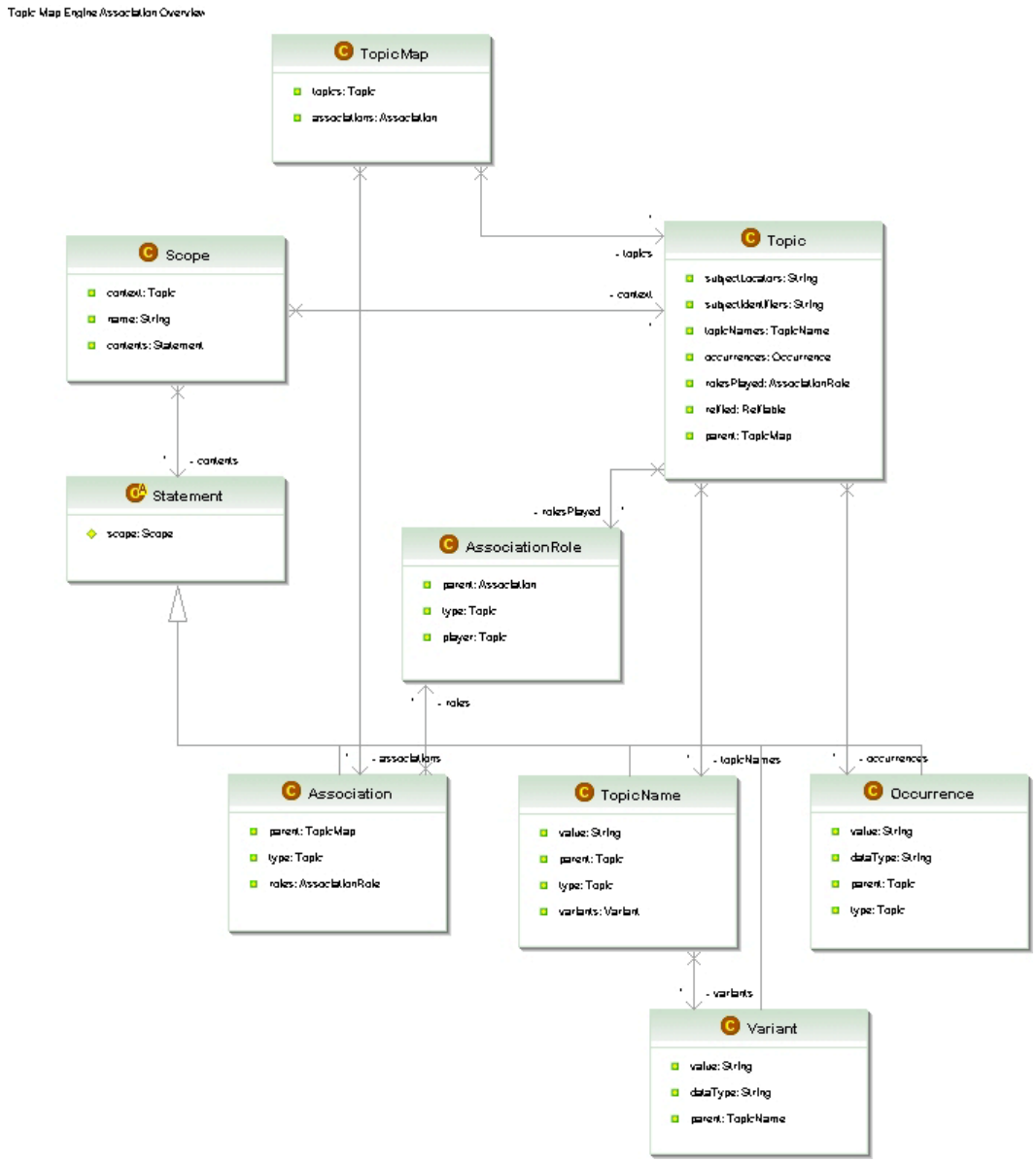

Figure 7.2: Topic Map Engine - Class Property & Association Overview

Chapter 8

Representation of OL Content

To boldly go where no man has gone before ...

from Star Trek - Intro

Objectives of Chapter 8

The goal of this chapter is to develop topic map representations of the models described in chapter 4. The development of management routines for these content models complements the basis for flexible OL content representation and management. All examinations are based upon the conceptual foundations of Scholion, in which the results of this work are integrated.

This chapter describes how the existing implementation of Scholion WB+ can be complemented by a topic map data layer and how Scholion can be extended with descriptions of procedural content. The first part of this chapter covers the concepts applied to express the models presented in chapter 4 to topic map constructs as described in annex B (and implemented as described in chapter 7). In the second part, the routines to access and manage OL content are presented - these routines make the topic map representation invisible for the user and enable convenient access. In the last part, the data import from Scholion is described. This module has only been implemented prototypically as it is out of scope of this work.

8.1 Mapping Content Models to Topic Maps

The OL Content Topic Map represents the models defined in chapter 4 in a semantic network. It decomposes the model elements into topics and associations, which are then used as topic types, association types, association role types or occurrence types in topic maps representing concrete content. The OL Management Layer (cf. section 8.2) uses the Content Topic Map to create and manage Scholion OL Content. The implemented content models are described in detail in annex D.

Means for Content Validity Checking

The OL Content Topic Map provides means of checking validity and soundness of represented content. Explicit definition of valid associations and roles for topic types enabled implementation of these features. The topic map standard does not provide any means of expressing these constraints - it is not possible to define validity rules fpr relationships between topics of a certain type, associations, association roles or occurrences. While the draft standard of the Topic Map Constraint Language [ISO05] is a candidate to close this gap, a different approach has been chosen. The Topic Map Constraint Language is not yet fully specified and has so far never been implemented - besides that, it introduces a completely new language for representation, for which no parser is currently available. The approach of this work is to express validity and soundness rules by means of topics and associations themselves.

A *Validity Meta Model* has been introduced and is used to specify the valid associations between concepts of the Scholion OL Content Topic Map (for an example see figure 8.1). Is is composed of the following elements:

- a topic type *AssocType*, which is used to type topics as association types in a concrete topic map
- a topic type *TopicType*, which is used to type topics as topic types in a concrete topic map
- a topic type *AssocRoleType*, which is used to type topics as association role types in a concrete topic map

- an association type *assocDef*, which is used to type associations that define valid combinations of association types and corresponding association role types. Associations typed with *assocDef* are always n-ary and are attached to the following topics:
  - exactly one topic typed with *AssocType* in the role *type* (which consequently only occurs once)
  - an arbitrary number of topics typed with *AssocRoleType* in the role of either *card_1* or *card_n* expressing the cardinality in which association roles (and implicitly attached topics) occur in the defined association.
- an association type *validRoleTopicCombination* used to type associations that define the valid combinations of association role types and topic types (i.e. which type of topic can take which roles in a certain association). Associations typed with *validRoleTopicCombination* are always binary and are attached to the following topics:
  - exactly one topic typed with *AssocRoleType* in the role *assocRole*
  - exactly one topic typed with *TopicType* in the role *topicType*

These elements are used to define each element and association given in the content models (in chapter 4). Extensions of these models always include a mapping to topic map representation by means of the *Validity Meta Model*.

Decoupling Content and exact Semantic Type

While the basic type of an element always remains constant, the actual meaning depends on the context, in which this element is used. This requirement was discovered in the course of translation from the content models to a topic map representation. It showed up when designing the mapping of the learning content model and was also found applicable to the process content model.

This led to a generic design decision: The basic type of an element is described as a topic type amd the exact semantics are assigned through roles (e.g. type: *block* - role: *example*, type: *WHAT* - role: *action*). In this way, the same content can be used with different meaning in different contexts. For instance, the *block* element in the learning content model can be semantically refined with

the types given in section 4.2 (e.g. as an example or a definition). Another example is the *WHAT* element of the process content model, which can be semantically refined e.g. to action or operation - depending on the actual context.

Concrete Mapping

A mapping of existing concepts for Scholion learning content to Topic Map constructs and structures has been developed (cf. Figure 8.1) in the course of several workshops with the Scholion development team. The validity meta model and models of learning and communication content (as described above) also resulted from these discussions.

Figure 8.1 shows the topic map representation of the content area. The same concepts have been applied to the communication model and the process model. Due to the rapidly rising complexity and size of the representation of larger models, both have not been visualized graphically but were implemented directly.

8.2 OL Management Layer

The OL management layer makes the topic map data representation invisible for users. It encapsulates both Scholion OL concepts and manipulation routines into comprehensible and easy-to-use classes and methods. In these classes, each concept and manipulation is translated into corresponding topic map concepts and operations, which are represented in and executed on the underlying topic map.

The topic map itself not only contains the actual content but also the model elements. So, every topic map is self contained from a data representation point-of-view.

Manager-classes provide access to the topic map. They have been implemented for every type of content and build a modular, layered structure, which encapsulates data representation. The result is an easy to use and extend framework for OL content representation (cf. Figure 8.2).

The managers provide the following functionality:

Topic Map Manager This manager encapsulates topic map details on the lowest level of abstraction. It provides means to manage topic and association

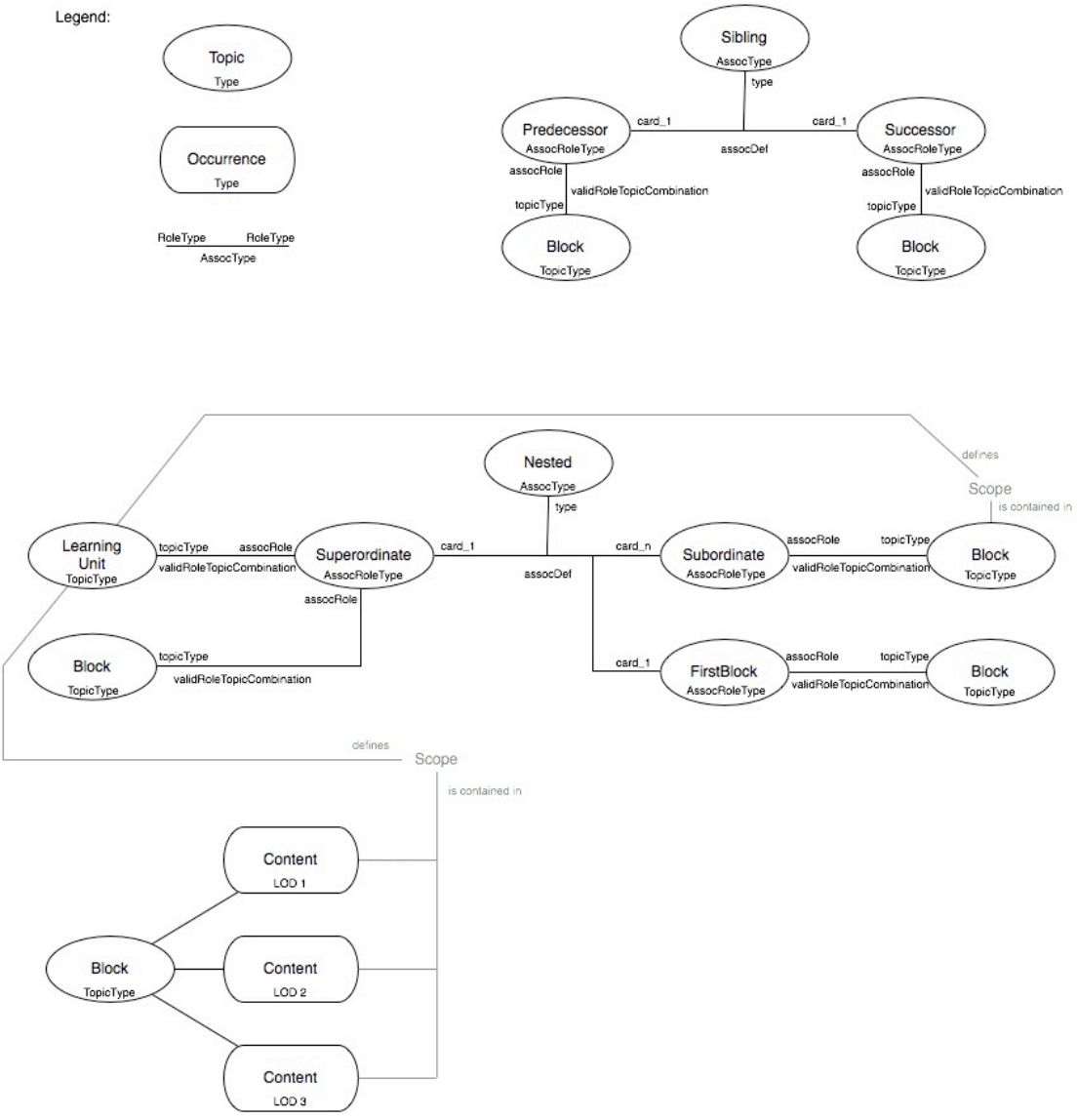

Figure 8.1: Representation of Scholion Content using TopicMap concepts

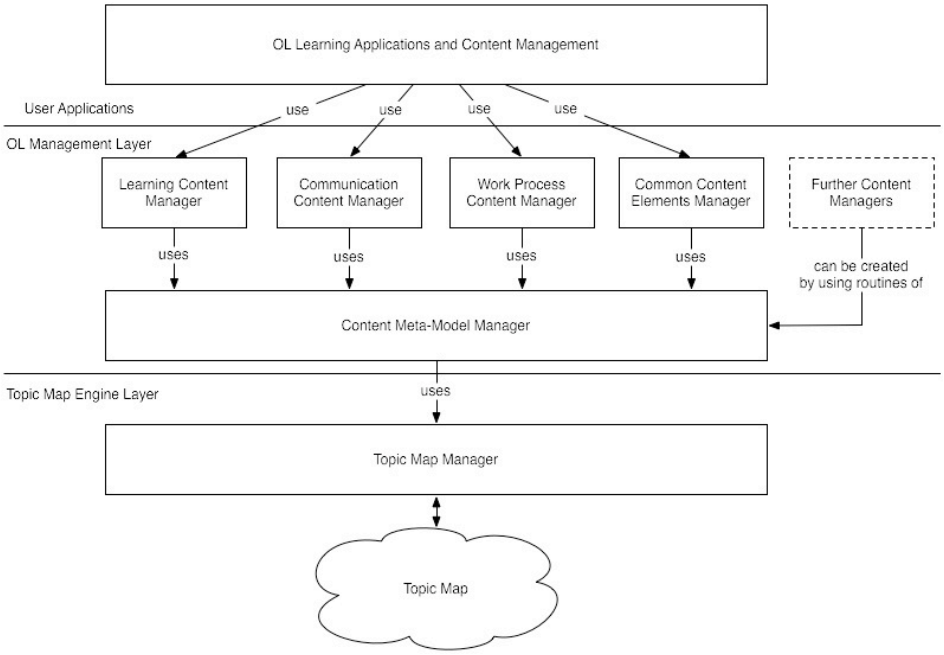

Figure 8.2: Structure of OL Management Layer

types, to define topics (with or without specified types), associate topics (in defined roles) and to manage scopes.

Content Model Manager The content model manager builds upon the topic map manager and provides means to define and manage content models for OL. It contains methods implementing the meta modeling concepts presented in subsection 8.1. These concepts are used to specify the elements contained in a content model and the valid associations (including roles). This manager class is the focal point for extension of the content models and is used to define (a) new elements and/or (b) new associations.

Learning / Communication / Work Process Content Manager The actual content manager classes are implemented for each content type supported in the OL platform. They encapsulate the topic map representation in an easy-to-use class interface. For every content element, a separate class is provided. By instantiating these classes new content elements are defined. Each element-class provides methods to establish links to related elements without having to care about the actual topic map representation (e.g. add blocks to a learning unit by calling `lu.add(set of blocks)`

and have the manager keep track of the necessary associations and scope definitions)

Common Content Elements Manager The common content elements manager contains means to manage both, elements that are used in each model and associations that span across distinct content models. Examples of elements that are used in every content model are *Course* (which is the basic container element in Scholion) and *Subject* (which represents humans either being users of the system, authors or even part of the content).

Every manager initialises itself by adding its elements to the topic map (including the topics and associations needed for representation of the (meta-)model it manages). In this way, every topic map contains all information necessary to reconstruct and navigate through the underlying models, that build the foundation for the represented content.

8.3 Scholion Data Importer

The *Scholion Data Importer* imports existing Scholion content into the topic map based representation format. The importer handles both, learning content and communication content (including infoboard, chat and forums). The Scholion Data Importer enables smooth transition from the old to the new, extended data management layer. It also facilitates parallel operation of both systems during time of transition.

The Scholion Data Importer reads Scholion Content from the Scholion database (cf. figure 6.1) and maps it to topic map representations (cf. section 8.2). It only works one-way and is not able to transfer content structured in topic maps back to the database representation (as the topic map representation model is a superset of the database schema in terms of relationships between content elements).

Chapter 8: Summary

In this chapter, the software modules necessary to complement Scholion with a topic map-based content data model have been developed. The results presented here can be considered a core result of this work and build the foundation the following chapters.

Part IV

Evaluation

Chapter 9

Content Structure Visualization as Enabler for Testing

...when thinking about ontologies and semantic web it is easy to focus on the requirements of precision and data integration to the exclusion of the requirements for end user navigation

Dave Reynolds et al. in [RSCS04]

Objectives of Chapter 9

A sample application for navigation in OL content based on topic maps is described in this chapter. One of the objectives in this work is to show applicability of the developed concepts in actual application scenarios. In addition, a means of structured testing of the software is needed. The application developed here sets on top of the OL Content Management Layer to meet these requirements. It visualizes content structures by exploiting the semantic information represented in the topic map.

The application presented here supports users to capture the context of a content element provided by Scholion. Several authors (e.g. [Bor04] or [CHKM05]) argue that learning in context has positive effects on successful knowledge transfer. The context in this case is constituted by all elements that are directly associated with the element of question. The semantic relationships between the

elements are derived from the underlying topic map representation and are visualized using the GraphViz toolset [GN00] (see Figure 9.1).

Figure 9.1: Example of Content Structure Visualisation

The current implementation generates static graphical visualisations of an element's context. A set of navigable HTML-pages is generated using the GraphViz toolset. This structure enables navigating through learning material by focusing on one content element at a time and providing hyperlinks to the directly attached elements. The focused element (in the center of figure 9.1) is augmented with a link to the actual learning content (automatically generated from the respective occurrences defined in the underlying topic map).

The visualiser currently is only a proof-of-concept prototype and suffers the following limitations:

- Semantics of associations are not considered during rendering, all associations are treated the same way (instead of e.g. marking type-instance-relationships differently).
- Multiple occurrences (e.g. for LOD1, LOD2 and LOD3 in learning content) are not handled, the focused element is always (and only) linked to the first occurrence.
- The content models have not been made navigable, although content type elements are displayed (as part of type-instance-relationships). Navigation is currently only possible on instance-level.
- Scopes are not considered during rendering, which leads to rather confusing visualisations for complex scenarios (e.g. for block-elements, which are included in several learning units).

However, the implemented application provides sufficient functionality to serve as a foundation for the user-test (see section 10.2). The visualiser uses all implemented components and in this way also serves as a testing instrument for formal functional evaluation (see section 10.1).

Chapter 9: Summary

In this chapter, an application to visualize the context of a content element has been developed. The information necessary for visualization is derived from the semantic data represented in the topic map. The application is a pointer on possibilities to generate added value for the user using the results of this work. Furthermore, it serves as the showcase used for the evaluation described in chapter 10. This chapter contributes to objective 6a.

Chapter 10

Evaluation Design

I think we ought always to entertain our opinions with some measure of doubt. I shouldn't wish people dogmatically to believe any philosophy, not even mine.

Bertrand Russell

Objectives of Chapter 10

This chapter contains descriptions of designing the tests and evaluations that are necessary to demonstrate the functionality and usefulness of the software developed in this work. Every component is tested formally, the overall system is evaluated in a user test. It is necessary to define required properties of the test content and criteria on which to judge the results.

Functional tests and user evaluation of the developed system are described in this chapter. The implemented modules are tested using an exemplary content structure defined in the following.

To test the central features of the software system on all layers (for details see section 10.1) the following requirements have to be fulfilled by the test content:

- Use elements from at least two different content models
- Use elements from the common content model

- Use model-internal content structures with at least to two links between at least two different content elements
- Have actual content linked to the elements, where content visibility has to be dependent on the scope (e.g. adding content to a block, which is only valid in one of two learning units)
- Use cross-content-model associations between content elements
- Use at least one sub-roletype to associate elements (see subsection 8.1)
- For learning content: Have at least one block elements reused in two learningunit elements

Designing test content following these requirements allows to test the topic map engine, covers large parts of the OL semantic content model and enables in-depth tests of the OL management layer. Furthermore, it is compatible with the current functionality of the visualizer application. The following structure has been developed to meet the given requirements:

- Course *Textbased Datamanagement*, containing two learningunits and a forum:
  - Learningunit *XML Basics*
  - Learningunit *XML Structuring*
  - Forum *XML Discussions*
- The learningunit *XML Basics* consists of the following blocks (all of the type *content*):
  - Introduction
  - Basic Structure
  - Structure Definition
    - DTD
    - Schema
- The learningunit *XML Structuring* consists of the following blocks:
  - Structure Definition (same block element as in the first learning unit)

  - DTD Example

- The forum *XML Discussions* contains a discussiontopic *XML Specifics*, which holds a discussion about how to *specify attributes*
- All blocks have been added a link to actual content. This content was marked to be *LOD2* in the context of the learningunit *XML Basics*.
- The block *Structure Definition* has been linked to the discussion about how to *specify attributes*

Based on this content structure, the functionality of the internal system components is tested. A *user evaluation* to test practical usage of the system includes (a) creating content structures and (b) using these structures for navigating (using the application presented in chapter 9).

10.1 Formal Tests

The formal tests described here show the correct *internal* operation of the developed system. In detail the following functionality is tested (affected modules given in italic font):

- Correct representation of topic maps from both, a structural and content point of view (*Topic Map Engine*)
- Correctly storing and retrieving topic maps to and from XTM and RDBMS (via Hibernate) (*Persistency Interface, XTM Codec, Hibernate Codec*)
- Correct generation and association of OL content elements (*OL Management Layer, OL Content Models*)
- Correct mapping of OL content elements to topic map constructs (*OL Content Models*)
- Correct visualization of content structures (*Content Structure Visualizer*)

The following tests have been designed to verify the areas given above (the criteria to be met are given in italic font):

- Correct representation of topic map

  - Visualize complete topic map (exemplary content instance including elements of all content models) and compare result to nominal structure
    Visualized structure contains all topics, associations and roles that have to be contained following the nominal content models and mappings to topic map structures given in chapter 4 and chapter 8

- Persistency (storing and retrieving topic maps)
  - Store Topic Map in XTM File
    XTM file has to be valid against the official XTM schema, all topic, associations and occurrences have to be stored
  - Retrieve Topic Map from XTM File
    The restored topic map has to be equivalent to the original one
  - Store Topic Map using Hibernate
    All topics, associations and occurrences have to be stored
  - Retrieve Topic Map using Hibernate
    The restored topic map has to be equivalent to the original one

- Generation and association of content elements
  - Generate a content instance for every implemented content model
    Creation and intra-model association of content elements has to be possible exactly for the combinations defined in the models given in chapter 4
  - Generate inter-model links between elements
    Inter-model association of content elements has to be possible exactly for the combinations defined in the models given in chapter 4

- Mapping of content elements to topic map representations
  - Visualize complete topic map (exemplary content instance including all content models) and context of all contained elements, compare results to nominal structure
    Visualized structure contains all topics, associations and roles that have to be contained following the definition of the exemplary content

- Visualization of content structures
  - Visualize the context of all elements contained in the example content structure
    All elements and associations have to be visualized as defined

10.2 User Evaluation

The user evaluation described here examines the applicability of the developed system in daily use scenarios. Focusing on applicability during design time, the scenario of creating content structures is used for testing.

A person familiar with the creation of learning content in Scholion is given the task to transfer the structure of some existing Scholion learning content to the OL platform (creation of two full learning units and a forum (including some discussions), which then are assembled to a course).

Chapter 10: Summary

This chapter defines the frame of reference for both, formal testing and user evaluation and enables accomplishment of test and structured verification of the results. The results of the tests and evaluations defined here are given in chapter 11.

Chapter 11

Evaluation & Results

1. If reproducibility may be a problem, conduct the test only once.
2. If a straight line fit is required, obtain only two data points.

Velilind's Laws of Experimentation

Objectives of Chapter 11

In this chapter, the tests and evaluations carried out with the developed software system and their results are described. The goal is to find out, whether the system operates correctly and if it is of principle use for users.

The formal tests and their results are presented in the first part of this chapter. In the second part, the user evaluation and its result is reflected briefly.

11.1 Formal tests

Following the requirements and criteria defined in section 10.1, the test of the overall software system and the contained components have been carried out. The content structure defined in chapter 10 has been used for these tests. It contains elements of all implemented content models and is designed in a way, which covers all relevant functionalities and specifics of the underlying software components. The results of the tests are described in the following:

Correct representation of topic map The complete topic map structure was rendered to test the correct representation of the topic map data. The output was then mapped to the structure defined by the content models and the test content structures. Figure 11.1 shows the visualisation of the topic map, which has been reflected (actually part by part, each model at a time) against the expected structures.

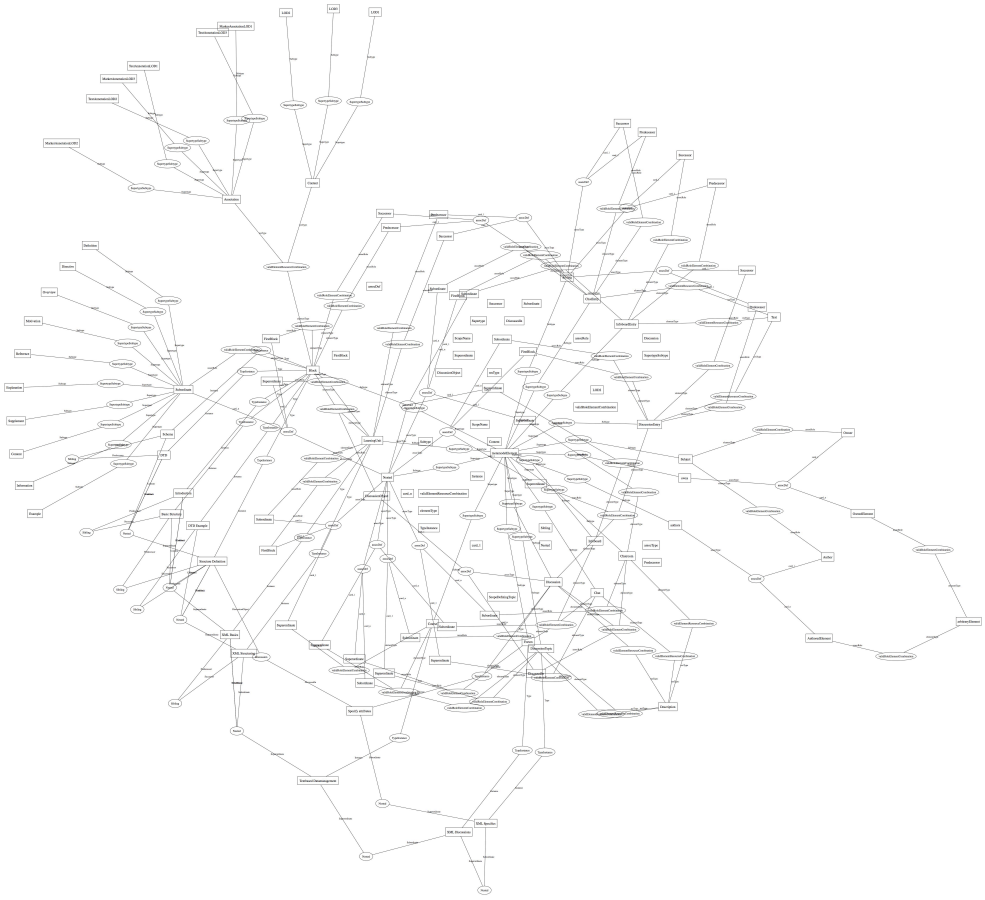

Figure 11.1: Visualisation of complete topic map structure

The mapping to the specified structures showed, that the topic map structures correctly represent content. Hence, the criteria are fulfilled for this test.

Persistency The exemplary content structure has been mapped to topic map representation for this test. The topic map was then rendered using the XTM and Hibernate Codec. Output worked as expected, the resulting XTM-file is valid

against the XTM Schema, the data structures in the relational database contain all information of the topic map (actually a 1:1 mapping between tables and objects has been created). Input also works as expected, the topic map structures is restored correctly (verified using a visualization as described above). However, the routines to restore the content element structures are not fully operable in the current implementation and were not tested in depth. Therefore, the criteria of this test are partially fulfilled, lacking full verification of restoring content structures from external sources.

Generation and association of content elements Test of content element generation and association was conducted using the exemplary content structure and some additional elements to cover all implemented features. The tests included generating instances of elements, associating them using the implemented service methods (cf. annex D) and retrieving them again. Correct operation has been verified with verbose console output of debug messages. All implemented element and manger classes operate correctly.

Mapping of content elements to topic map representations The mapping routines from OL content elements to topic map representations were tested for all implemented elements and the exemplary content structure. Representation of content elements used multiple times in different contexts (and in different roles) was also tested. The topic map representations of the test structures work as expected (after some revisions).

Mapping to topic maps of content structures has been implemented by calling the mapping routines of each contained elements recursively (by traversing the associations between the elements). This approach still it contains potential for improvements of efficiency. The mapping routines of elements with multiple connections to their environment are called several times now, which has negative impact in performance in larger structures. However, this does influence correct operation.

A lack of functionality has been identified for generating and associating elements from underlying topic map representation (as already mentioned before). Element generation works, however, reconstructing the associations between elements is still under development.

Visualization of content structures The visualisation application was tested by rendering the whole content structure including content models (to test completeness of visualization) and by rendering the context of every element contained in the exemplary content (to test comprehensibility of visualisation and linking to content and other element's context). After some tuning of the parameters of the visualization engine, both tests have been completed successfully. Based upon this, comprehensibility was tested in detail in the user evaluation described in the next session.

11.2 User evaluation

A member of the Scholion development team was asked to evaluate the management routines of the system by creating exemplary learning content. Creation was carried out by invoking the service routines directly from java code (within a test driver). Both, the learning content model and the communication content model, have been tested.

A learning content structure was created in the first step. The creation process was preceded by a short introduction to the structure of the management routines. The sample content covered all specified element and association types of the learning model.

In the second step, communication content was created and linked to learning elements. A forum containing topics and discussions was established. A selected discussion was then linked to a block element of the learning content.

Both creation steps were supported using the content structure visualiser, which displayed the already created elements and the established associations.

The following feedback on appropriateness of the system was received:

- The structure of learning content and communication content was mapped correctly to the built models.
- The management routines as a whole are understandable and appropriate for creating content structures.
- The mapping of Scholion elements to distinct classes allows easy composition of learning material.

- The inter-content-model linking routines (e.g. for linking discussions to learning content) work as expected. However identification of elements other than by their designators (e.g. by an id) would make management easier.
- The content structure visualiser helped to keep track of the content creation process.
- Assignment of learning-unit specific block types, content representations, and annotations is rather complicated to use but implemented in a sound, comprehensible way.
- Management of annotations already works as expected but needs more sophisticated management routines (especially in terms of altering annotations).
- Detailed knowledge on topic maps is not needed to use the system.

The user evaluation did not identify any previously unknown shortcomings of the system. However, it confirmed the already known areas for improvement. Overall, the system was considered appropriate and usable for representation of Scholion content.

Chapter 11: Summary

This chapter has described the accomplishment of the tests and evaluations as defined in chapter 10. The results show, that the main objectives of the work have been fulfilled. However, certain deficiencies in terms of functionality and stability have shown up, which will have to be corrected in future work (see section 12.3). This chapter completes the work on objectives 5e and 6b.

Chapter 12

Conclusions

'Where shall I begin, please your Majesty?' He asked.
'Begin at the beginning,' the King said, very gravely,
*'and go on till you come to the end: **then stop**.'*

The White Rabbit & the King in [Car66]

The global goal of this work was to design and develop a system which enables flexible representation of OL content. The main objectives have been:

- Identify the requirements OL approaches and eLearning concepts set on data representation for content.
- Identify and justify the kinds of content (content types) necessary for organisational learning.
- Develop models for every content type, which provide a respective frame of reference for each area (i.e. define the types of elements and how they can be associated). This is necessary to build a meshed content structure, which provides semantic added value.
- Design and implement a software system, which allows to manage these content structures by the means of suitable data representation. The user of this software (e.g. content developers), however, must not be burdened with managing representation interna but have to be provided a simple, content-oriented interface for building content structures.

- Build a sample application, which uses the developed software components. This application is used for testing the components as well as demonstrating the potential of this approach for users.

Each of these objectives has been reached. Some components have to be considered preliminary, prototypically or under development as regards implementation. However, the principle functionality has been demonstrated in the current state of implementation.

12.1 On the Use of Topic Maps

Topic maps have been chosen to be the conceptual and technical foundation for this work. With their generic approach to data representation, they are powerful means to represent content. Content can be flexibly assembled, intertwined and instantiated using different information representations depending on the current learning scenario.

The concepts of association roles, occurrence types and scopes enable to change the meaning and actual appearance of a content element depending on the context it is used in. In daily use, learning content can be reused for different learning contexts and can be augmented by user's connotations (i.e. define the individual meaning of a content element). Content can be instantiated in arbitrary types of representation and level of detail by using occurrences. It is even possible to change the representation of content depending on the current learning context.

Using type-instance-relationships in a topic map, it is possible to represent content models in topic map constructs and in this way integrate them directly into the topic maps. Thus, every topic map is self contained, i.e. besides the actual content, it also contains the models necessary to interpret this content and verify its structure.

Currently, it is not possible to formulate constraints in a standardized way (i.e. which types of topics can be associated by which types of associations in which roles, etc.). A language to specify constraints - TMCL (Topic Map Constraint Language) - is currently developed to remove this limitation. However, TMCL is not fully defined yet - support can be hardly implemented at the moment. For this work, the formulation of constraints was implemented using specific associations, which have been defined in a separate meta-model. Knowing

how to interpret these meta-associations enables to check the validity of the represented content structure. Together with the self-containedness of representation, powerful means of data exchange are created. Models for new content types can be easily distributed together with the actual content.

All this flexibility leads to high complexity of the resulting models. It is unacceptable to confront end users with the entire topic map structure of even the simplest learning content (as already the content models consist of about 100 elements and several hundred associations). Hence, the topic map constructs were encapsulated in content element wrappers, which are easy and straightforward to use. However, to keep the advantages of the topic map representation in terms of semantic association and context queries, users are also given direct access to the topic map layer. Manager classes have been developed for each layer of the software system (on topic map layer, meta-model layer and model layer) in order to provide structured and governed access to these components.

In all, topic maps have proven the ideal (and so far only) means to manage content structure in the required polymorphism and interconnectedness. However, the complexity that goes along with their representational power makes topic maps hard to manage and requires sophisticated routines for access and modification.

12.2 On relevant Schools of Knowledge Management

In this section, this work is put into the context of approaches and schools of knowledge management (KM). The main objective of this work - to create a flexible means of data representation for OL content - can only be fulfilled when the different schools of KM and OL are considered in this work. The review of existing OL approaches (which actually also contained approaches commonly considered to focus on KM) was a first step to meet this requirement. In this examination, it became obvious, that by far not all approaches substantially contribute to the foundations of this work.

Candidates for KM approaches of interest are those which explicitly focus on knowledge as a (tangible or intangible) good that can be transfered between individuals. Furthermore, those approaches are of interest, which give statements

on how knowledge transfer happens. These constraints originate from the assumption, that organisational knowledge actually is knowledge of individuals which is share and spreads across the whole organisation.

Following these constraints, two schools of KM are considered relevant: the SECI approach of Nonaka and Takeuchi (and the works building upon them, like Krogh et. al.) and the Knowledge Lifecycle of Firestone and McElroy (and their conceptual predecessors, Argyris & Schön as well as Kim).

The SECI approach is a conceptual framework of how knowledge is shared among individuals. It differentiates between implicit and explicit knowledge and identifies four basic means of knowledge transfer: socialization, externalisation, combination and internalisation. When putting this work in the context of these four means, it obviously focuses on the latter three, where the central point is support for combination of explicit knowledge. Besides that, the approach presented here aids individuals to externalize their view on work processes (externalisation) and helps them to understand that of others in order to develop a common view on their work (internalisation). This is however out of scope of this work and will be elaborated and examined in future work.

The Knowledge Lifecycle describes how organisational knowledge is created and used. Firestone and McElroy put the process of knowledge creation in the context of an organisation's business processes. This work supports a similar assumption by stating, that organisational learning happens along daily work processes. The knowledge lifecycle therefore provides an explanatory model of the stages to be supported during learning using a platform that supports OL. Again, the organisational learning process itself is out of scope of this work and will have to be examined in future research.

12.3 On Directions for further Development

The work presented here has to be seen as a first step towards the development of a platform to support organisational learning. In combination with the existing implementation of Scholion, the software results of this work provide the foundations of this platform. The topic map engine and the OL content models (together with their access routines) build the future data management layer of Scholion and will allow new applications and features to support learning along the work process.

Conceptually, this work has reached a stable state and can be used as a foundation for further development. From a software point of view, several components have to be completed and/or stabilized in order to get beyond prototype status:

- Complete the content model for work-process content (which currently has only been designed conceptually)
- Perform in-depth tests of the software components with real-world learning content
- Integrate the new data management layer with Scholion, starting with the implementation of a Scholion Data Importer to transfer existing learning content structures to topic map representation
- Redesign and refine the application for context visualization of learning content.

Most of these points are already subject of starting or ongoing project work and diploma thesis supervised by the author of this work. The goal is to have complete and fully functional prototype of the software system (including the points given above) by the end of 2007.

Data representation based upon topic maps opens up a whole new world of possibilities for individualized learning support in educational and organisational settings. Providing individual learning context and means for interaction among individuals (in its broadest sense) are the first steps towards computer supported learning settings, which may sensibly support or even come close to learning in actual organisational scenarios.

Bibliography

[All02] V. Allee. *The Future of Knowledge: Increasing Prosperity Through Value Networks*. Butterworth-Heinemann, 2002.

[Arg90] C. Argyris. *Overcoming organizational defenses, facilitating organizational learning*. Allyn and Bacon, Boston, 1990.

[AS78] C. Argyris and D. Schön. *Organizational Learning: A Theory Of Action Perspective*. Addison-Wesley, 1978.

[AS99] C. Argyris and D. Schön. *Die lernende Organisation: Grundlagen, Methode, Praxis*. Klett-Cotta, Stuttgart, 1999.

[AS05a] A. Auinger and C. Stary. *Didaktikgeleiteter Wissenstransfer. Interaktive Informationsräume für Lern-Gemeinschaften im Web*. Deutscher Universitätsverlag, 2005.

[AS05b] A. Auinger and C. Stary. Effektive Content-Produktion für selbstgesteuerten, polymorphen Wissenstransfer. In *Proceedings of 7. Internationale Tagung Wirtschaftsinformatik (WI2005)*, 2005.

[Aui03] A. Auinger. *Technologische Unterstützung didaktikgeleiteten Wissenstransfers*. PhD thesis, University of Linz, Oct., 2003.

[AvE01] A. Abecker and L. van Elst. Integrating task, role and user modeling in organizational memories. In *The 14 Int. FLAIRS Conference Proceedings*, Key West, Florida, 2001. AAAI Press.

[BB03] O. W. Bertelsen and S. Bødker. Activity theory. In J. Carroll, editor, *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*. Morgan Kaufman Publishers, 2003.

[Bor04] N. Boreham. Orienting the work-based curriculum towards work process knowledge: a rationale and a German case study. *Studies in Continuing Education*, 26(2):209--227, 2004.

[Car66] L. Caroll. *Alice's Adventures in Wonderland*. D. Appleton & Co, New York, 1866.

[Car71] L. Carroll. *Through the Looking Glass*. D. Appleton & Co, 1871.

[CE93] M. Cole and Y. Engeström. A cultural-historical approach to distributed cognition. In G. Salomon, editor, *Distributed cognitions. Psychological and educational considerations*, pages 1--47. Cambridge University Press, 1993.

[CHKM05] A. Carell, T. Herrmann, A. Kienle, and N. Menold. Improving the coordination of collaborative learning with process models. In T. Koschmann, D. Suthers, and T.W. Chan, editors, *Proceedings of CSCL 2005. The next 10 Years.*, pages 18--27, 2005.

[Cre05] A. Cregan. Building Topic Maps in OWL-DL. In *Proceedings of Extreme Markup Languages 2005*, Montreal, Canada, 2005. IDEAlliance.

[DR97] C. Dahme and A. Raeithel. Ein tätigkeitstheoretischer Ansatz zur Entwicklung von brauchbarer Software. *Informatik-Spektrum*, 20:5--12, 1997.

[Eng87] Y. Engeström. *Learning by expanding*. Orienta-konsultit, Helsinki, 1987.

[Eng05] Y. Engeström. *Developmental work research: Expanding activity theory in practice*. Lehmanns Media, Berlin, 2005.

[Eul98] S. Eulgem. *Die Nutzung des unternehmensinternen Wissens*. Peter Lang Verlag, Frankfurt, 1998.

[FHL+98] E. D. Falkenberg, W. Hesse, P. Lindgreen, B. E. Nilsson, J. L. H. Oei, C. Rolland, R. K. Stamper, F.J.M. van Assche, A.A. Verrijn-Stuart, and K. Voss. A framework of information system concepts.

The FRISCO Report. online version, International Federation for Information Processing WG 8.1, 1998.

[FM03a] J. Firestone and M. McElroy. *Excerpt from The Open Enterprise: Building Business Architectures for Openness and Sustainable Innovation*. KMCI Online Press, 2003.

[FM03b] J. Firestone and M. McElroy. *Key Issues in the new Knowledge Management*. Butterworth-Heinemann, 2003.

[GN00] E.R. Gansner and S.C. North. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11):1203--1233, 2000.

[GR00] P. Green and M. Rosemann. Integrated process modeling: An ontological evaluation. *Information Systems*, 25(2):73--87, 2000.

[Gul05] A.D. Gulbrandsen. Conceptual modeling of topic maps with orm versus uml. In L Maicher and J. Park, editors, *Charting the Topic Maps Research and Applications Landscape. First International Workshop on Topic Maps Research and Applications, TMRA 2005.*, pages 93--106, Berlin, Germany, 2005. Springer.

[Hub91] G. Huber. Organizational learning: The contributing processes and the literatures. *Organization Schiences*, 2(1), 1991.

[ISO02] ISO JTC1/SC34/WG3. ISO/IEC 13250 Topic Maps. Standard 13250, ISO/IEC, May 2002.

[ISO05] ISO JTC1/SC34. Topic Maps Constraint Language. working draft standard, ISO/IEC, 2005.

[ISO06a] ISO JTC1/SC34/WG3. Information Technology - Topic Maps - Part 2: Data Model. International Standard 13250-2, ISO/IEC, June 2006.

[ISO06b] ISO JTC1/SC34/WG3. Information Technology - Topic Maps - Part 3: XML Syntax. International standard, ISO, June 2006.

[JRH05] I. Jahnke, C. Ritterskamp, and T. Herrmann. Sociotechnical roles for sociotechnical systems: a perspective from social and computer science. In *2005 AAAi Fall Symposium, American Association for Artificial Intelligence, 8. Symposium: Roles, an interdisciplinary perspective*. AAAI Press, 2005.

[KH04a] A. Kienle and T. Herrmann. Collaborative learning at the workplace by technical support of communication and negotiation. In *Multikonferenz Wirtschaftsinformatik (MKWI) 2004*, volume 1, pages 43--57, 2004.

[KH04b] A. Kienle and T. Herrmann. Konzepte für die Lerngruppe: Prozessunterstützung, Annotationen und Aushandlung. In J. Haake, G. Schwabe, and M. Wessner, editors, *CSCL-Kompendium*, pages 171--183. Oldenbourg Verlag, München, 2004.

[Kim93] D.H. Kim. *A Framework and Methodology for Linked Individual and Organisational Learning: Applications in TQM and Product Development*. PhD thesis, Sloan School of Managment, Massachusetts Institute of Technology, 1993.

[Kim01] D.H. Kim. *Organizing for Learning: Strategies for Knowledge Creation and Enduring Change*. Pegasus Communications, 2001.

[KNM99] V. Kaptelinin, B.A. Nardi, and C. Macaulay. Methods & tools: The activity checklist: a tool for representing the `space' of context. *interactions*, 6(4):27--39, 1999.

[LB98] H. Linger and F. Burstein. Learning in organisational memory systems: An intelligent decision support perspective. In *Proceedings of the 31st Hawai Conference on System Sciences*. IEEE Press, 1998.

[Leo78] A.N. Leont'ev. *Activity, Consciousness, and Personality*. Prentice-Hall, 1978.

[LLN00] T. Le, L. Lamontagne, and T. Nguyen. A visual tool for structuring and modeling organizational memories. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 258--263, New York, NY, USA, 2000. ACM Press.

[MDZH00] P. Mulholland, J. Domingue, Z. Zdrahal, and M. Hatala. Supporting organisational learning: an overview of the ENRICH approach. *Information Services and Use*, 20(1):9--23, 2000.

[MO82] J. March and J. Olsen. *Ambiguity and Choice in Organizations*. Universitetsforlaget, Bergen, Norway, 1982.

[MPS02] G. Mori, F. Paternò, and C. Santoro. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Transactions on Software Engineering*, 28(9):797--813, 2002.

[MRRvdA06] J. Mendling, J. Recker, M. Rosemann, and W. van der Aalst. Generating correct epcs from configured c-epcs. In H. M. Haddad, editor, *Proceedings of the 21st Annual ACM Symposium on Applied Computing*, pages 1505--1510, Dijon, France, 2006. ACM Press.

[Nar96] B.A. Nardi. Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition. In B.A. Nardi, editor, *Context and Consciousness: Activity Theory and Human-Computer Interaction*, chapter 4, pages 69--102. MIT Press, 1996.

[NT95] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[Ont03] Ontopia. The RTM RDF to topic maps mapping, http://www.ontopia.net/topicmaps/materials/rdf2tm.html, 2003.

[Onto6] Ontopia. RDF support in the Omnigator, http://www.ontopia.net/omnigator/docs/navigator/userguide.html, 2006.

[Paw01] J.M. Pawlowski. *Das Essener-Lern-Modell (ELM): Ein Vorgehensmodell zur Entwicklung computerunterstützter Lernumgebungen*. PhD thesis, University of Essen, 2001.

[PC05] J. Park and A. Cheyer. Just for me: topic maps and ontologies. In L. Maicher and J. Park, editors, *Charting the Topic Maps Research and Applications Landscape. First International Workshop on Topic Maps Research and Applications, TMRA 2005.*, pages 145--59, Berlin, Germany, 2005. Springer.

[Pep00] S. Pepper. The tao of topic maps. In *Proceedings of XML Europe*, 2000.

[Pol03] P.R. Polsani. Use and abuse of reusable learning objects. *Journal of Digital Information*, 3(4), 2003.

[Raf05] S. Raffeiner. Modelling ontologies with topic maps and owl: Implementation challenges and conceptual issues. Master's thesis, Technical University of Vienna, 2005.

[Ram96] B. Ramesh. Toward a meta-model for representing organizational memory. In *Proceedings of the 29st Hawai Conference on System Sciences*. IEEE Press, 1996.

[Rat03] H. Rath. *The Topic Maps Handbook*. empolis GmbH, 2003.

[Red07] Red Hat Middleware. Hibernate Reference Documentation. Reference documentation, Red Hat Middleware, 2007.

[RFW] M. Robertson, A. Fluck, and I. Webb. Activity theory.

[RR06] M. Rosemann and J. Recker. Context-aware process design: Exploring the extrinsic drivers for process flexibility. In T. Latour and M. Petit, editors, *The 18th International Conference on Advanced Information Systems Engineering. Proceedings of*

Workshops and Doctoral Consortium., pages 149--158, Luxembourg, 2006. Namur University Press.

[RSCS04] D. Reynolds, P. Shabajee, S. Cayzer, and D. Steer. Swad-europe deliverable 12.1.7: Semantic portals demonstrator- lessons learnt, 2004.

[SAJ+02] E. Soderstrom, B. Andersson, P. Johannesson, E. Perjons, and B. Wangler. Towards a framework for comparing process modelling languages. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering, CAiSE*, pages 600--611. Springer, 2002.

[Sch05a] C.O. Scharmer. Theory u: Leading from the emerging future - presencing as a social technology of freedom. Author's excerpt of forthcoming book, 2005.

[Sch05b] S. Schluep. *Modularization and structured markup for web-based learning content in an academic environment*. Shaker, Aachen, 2005.

[Sen90] P. Senge. *The Fifth Discipline: The Art and Practice of the Learning Organization*. Doubleday/Currency, 1990.

[SKR95] P. Senge, A Kleiner, and C. Roberts. *The fifth discipline fieldbook: strategies and tools for building a learning organization*. Nicholas Brealey, 1995.

[Sow84] J. Sowa. *Conceptual Structures*. Addison-Wesley, Reading, 1984.

[Sow00] J. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks-Cole, Pacific Grove, 2000.

[SSJF04] P. Senge, C.O. Scharmer, J. Jaworski, and B.S. Flowers. *Presence: Human Purpose and the Field of the Future*. Society for Organizational Learning, 2004.

[Ste89] E. W. Stein. *Organizational Memory: Socio-technical Framework and Empirical Research*. PhD thesis, University of Pennsylvania, 1989.

[Ste95] E. W. Stein. Organizational memory: Review of concepts and recommendations for management. *International Journal of Information Management*, 15(1):17--32, 1995.

[Sto03] S. Stoiber. *Organisationales Lernen 'Online' - Methodik und Werkzeug*. PhD thesis, University of Linz, 2003.

[Top01] TopicMaps.org Authoring Group. Xml topic maps (xtm) 1.0. Standard, TopicMaps.org, 2001.

[Vat04] B. Vatant. Ontology-driven topic maps. In *Proceedings of XML Europe 2004*, Amsterdam, 2004.

[VK00] J. Virkkunen and K. Kuutti. Understanding organizational learning by focusing on activity systems. *Accounting, Management and Information Technologies*, 10(4):291--319, 2000.

[vKNI00] G. von Krogh, I. Nonaka, and K. Ichijō. *Enabling Knowledge Creation: How to Unlock the Mystery of Tacit Knowledge and Release the Power of Innovation*. Oxford University Press US, 2000.

[vW01] M. van Welie. *Task-Based User Interface Design*. PhD thesis, Vrije Universiteit - Dutch Graduate School for Information and Knowledge Systems, 2001.

[W3C04a] W3C. Owl web ontology language overview, 2004.

[W3C04b] W3C. Rdf primer, 2004.

[W3C06] W3C. W3c rdf and owl activities, 2006.

[Wen99] E. Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge University Press, 1999.

[Wor95] Workflow Management Coalition. The workflow reference model. Technical report, Workflow Management Coalition, 1995.

[WW97] C. Wargitsch and T. Wewers. Workbrain: Merging organizational memory and workflow management systems. In *Workshop of*

Knowledge-Based Systems for Knowledge Management in Enterprises at the 21st annual German Conference on AI (KI-97), 1997.

Annex

Appendix A

Approaches to Organisational Learning

This annex contains a review of recent and historical approaches for organisational learning and organisational memories. All presented concepts have been reviewed regarding types of content and requirements on data representation. In the descriptions of the concepts, statements are given on *what to learn* and on the *required context of learning*. Both aspects contribute to the identification of the relevant types of content. Requirements on data representation are derived from statements on context of learning primarily. The results of the review are consolidated in chapter 2. The following concepts have been considered:

- Concepts focusing on the learning process
  - Cycle of Choice (March & Olsen)
  - Argyris & Schön
  - Huber
  - SECI-Model (Nonaka, Takeuchi & Krogh)
  - The Fifth Discipline (Senge)
  - Kim
  - Stoiber
  - ENRICH (Mulholland et al.)
  - Theory U (Scharmer, Senge, Jaworski & Flowers)
  - Knowledge Lifecycle (Firestone & McElroy)
  - Value Networks (Allee)
- Concepts focusing on the objects of learning

- Stein
- Abecker & van Elst
- Eulgem
- Linger & Burstein
- Ramesh
- Le, Lamontagne & Nguyen
- Wargitsch & Wewers

This annex complements chapter 2 with details on the reviewed approaches.

A.1 Concepts focusing on the Learning Process

A.1.1 March & Olsen

In the Cycle of Choice [MO82], March & Olsen adress four aspects which - when intertwined - constitute the foundations for organisational development: individual beliefs, individual actions, organisational actions and environmental response. They also define learning obstacles (role constrained learning, audience learning, superstitious learning and learning under ambiguity) which cause incomplete learning cycles by breaking the cycle between any two of the aspects mentioned before.

They do not give any statements about how learning itself occurs (no statement on *what to learn*). However, they give factors, which hamper learning (but do not give hints on how these obstacles could be overcome). The learning obstacle *Learning under ambiguity* allows certain assumptions on the *context* to be provided for effective learning can be derived. *Learning under ambiguity* occurs, whenever individuals lack enough information to interpret their environment (or its changes) and decide about their reactions correctly. This obstacle could be avoided by offering individuals information relevant in their current situation in a structured way (context-sensitive offering of information).

A.1.2 Argyris & Schön

Argyris & Schön [AS78, Arg90, AS99] are the first to distinguish between single-loop and double-loop learning. In *single-loop learning*, individuals or organizations (which are represented by individuals in this approach) modify their actions based upon the difference between expected and observed outcome. In *double-loop learning*, individuals question the underlying assumptions, values

and policies of an action and modify these. They introduce the concepts of *private images* and *organisational maps* and are the first to distinguish between individual mental models (or 'theories-in-use') and official organisational points-of-view (or 'espoused theories').

In [AS99], Argyris & Schön give statements on *what to learn* in OL. They aregue for information about *activities*, *decisions* and *procedure models*, i.e. information about the process within an organisation, irrelevant wether it is represented within documents or in people's minds. Regarding the *context* of OL, the authors state, that OL occurs, whenever people experience a mismatch between expected and observed results of their behaviour. They claim that integration into the organisational knowledge base (the 'pictures of the organisation') is inevitable to make organisational knowledge persistent - in both people's heads and organisational artefacts (e.g. diagrams).

A.1.3 Huber

Huber [Hub91] sees OL as process, consisting of knowledge acquisition (basically individual learning from practice), information distribution and information interpretation (basically learning from preprocessed information). Huber is one of the first to introduce the idea of an *organisational memory*, which makes information persistent.

Huber gives some hints on *what to learn* in his description of organisational memories. He distinguishes between soft facts (which are only available in people's heads) and hard facts (which include descriptions of routines, guidelines or organisational manuals). In his elaboration on information interpretation, Huber gives statements on the *context* of learning: existing individual mental models have to be considered during learning, media richness is a central point in efficiency of interpretation, information overload has to be avoided.

A.1.4 SECI-Model (Nonaka, Takeuchi & Krogh)

The SECI-Cycle of Nonaka and Takeuchi [NT95] and the extension of their work by Krogh et al. [vKNI00] provides explanatory approaches of how knowledge evolves and is shared (through socialisation, externalisation, combination and internalisation) and how knowledge creation can be enabled (with five identified enablers, which have to be implemented on strategical level).

In terms of *what to learn*, Nonaka and Takeuchi distinguish explicit and implicit knowledge. Explicit knowledge is somehow codified in artifacts and can easily be transfered. Implicit knowledge is only contained in people's heads, highly context-sensitive and hard to be externalised. They distinguish *cognitive elements* (conceptual understanding) and *technical elements* (know-how) of knowledge. Krogh et al.'s work gives some hints on the learning *context*: in order to enable knowledge creation - or learning - they suggest to set measures to *manage conversations* and to *create the right context* as well as to *globalize local knowledge*. While the authors give application examples in the individual, interpersonal area, their suggestions may also be applicable in the context of an electronic OL platform.

A.1.5 The Fifth Discipline (Senge)

In his most influential work [Sen90], Senge presents five disciplines, which have to be mastered to overcome obstacles on the way towards a 'learning organisation': *personal mastery*, *mental models*, *shared vision*, *team learning* and *systems thinking*. Senge mainly focuses on soft factors in his work, which are hardly relevant for this work and will not be presented here in detail.

However, some of the concepts he presents in the fifth discipline fieldbook [SKR95] provide input for the question on the appropriate *context* of learning: In the area of mental models, he proposes to provide means to express individual motivators and conceptual understandings and to explore those of others. In systems thinking, means to express causal dependencies (like causal loop diagrams) are considered useful to aid mastery of this discipline. As Senge focuses on the process to a learning organisation, he consequently gives no statements on *what to learn*.

A.1.6 Kim

Kim's framework for organisational learning [Kim93, Kim01] incorporates the approaches of March & Olsen [MO82], Argyris & Schön [AS78] as well as some ideas of Senge [Sen90]. Basically using the single- and double-loop-learning approach and extending it beyond individual to organisational level, Kim proposes to consider *mental models* to be the focal subject of organisational learning.

Mental models contain *framework*, i.e. know-why or conceptual understanding of a phenomenon, and *routines*, i.e. know-how or procedural knowledge.

Frameworks and routines as part of mental models are also the central element Kim gives in terms of *what to learn*. In terms of the learning *context*, Kim adds some aspects to March & Olsen's learning obstacles. *Situational Learning*, which occurs when new knowledge is not made persistent and *fragmented learning*, which occurs when new knowledge does not spread across the organisation, point out the need for a means of external knowledge representation and distribution.

A.1.7 Stoiber

The approach of Stoiber [Sto03] focuses on business processes as a means for and subject of organisational learning. The work presented here was largely inspired by this approach. Stoiber has examined the role of representations of work processes during organizational learning (in the understanding of Kim [Kim93]).

In terms of *what has to be learned*, Stoiber identifies knowledge about work processes to be a central element, to which further information can be attached or incorporated, respectively. In terms of *context* of learning, Stoiber states that for successful OL, learners have to be aware of the actual context of work (which itself can be subject of learning). She proposes to use graphical representations of business processes to enable work context awareness.

A.1.8 ENRICH (Mulholland et al.)

The ENRICH approach [MDZH00] presents a framework to support organisational learning. The starting point are *work representations*, which are resources (documents or tools) used during work and contain or represent knowledge crucial to the organisation. These work representations are then 'enriched' with means of communication and semantic links to related information.

Mulholland et al. do not say much about *what has to be learned*, i.e. which kind of information is exactly represented in work representations. In terms of the learning *context*, they state, that OL happens in the context of actual work, which has to be provided to the worker during learning - in this case, this is realized by providing the respective work representations.

A.1.9 Theory U (Scharmer, Senge, Jaworski & Flowers)

Scharmer et al. [SSJF04, Sch05a] present a framework to explain and aid change in organisations - the Theory U. Theory U describes a process to shift and deconstruct perspectives until something 'new' arises which is subsequently stabilized and put to practice. The focal point of the process is the evolution of new and deeper insights and is called *Presencing*.

While Theory U seems to be a promising approach to aid processes of profound change, the authors remain vague in both, terms of *what is the subject of learning* and terms of the *context* of learning. Theory U therefore does not add any new aspects to the topics of interest in this review.

A.1.10 Knowledge Lifecycle (Firestone & McElroy)

In the Knowledge Lifecycle [FM03b] Firestone and McElroy present a process-oriented framework of how organisational knowledge creation and consolidation works (founded on single- and double-loop-learning-cycles). They put knowledge processing in the context of business processes and define a *distributed organisational knowledge base*, which represents all organisational information (where the central building blocks are beliefs and *knowledge claims*), regardless if it is contained in peoples' heads or codified in artifacts.

In terms of *what to learn*, the knowledge lifecycle explicitly distinguishes different types of information representation (individually or codified in artifacts) but does not give any statement on the types of information contained in these representations. In terms of *context* of learning, the starting point for production of new knowledge or consolidation of existing one always is an actual business process. Knowledge production is triggered by matches or (more likely) mismatches of expected and observed environmental reaction.

A.1.11 Value Networks (Allee)

Value Networks [All02] are an approach to describe organisations (or more generally economic systems) with focus on the internal and external exchange of values (which are either tangible or intangible). While value networks are a means to analyze both flows of material and immaterial goods, Allee does not explicitly focus on the learning aspect in value networks. However, her framework allows

to capture learning as an activity caused by the flow of values (like every other activity could be expressed).

In her book and subsequent publications, Allee gives examples, which cover aspects that are interesting for this review. In terms of *what to learn*, Allee specifies, that intangible values exchanged between nodes in the network include information about processes and structure as well as declarative information of any kind. In terms of the *context* of learning, Allee proposes graphical visualisation methods in order to facilitate understanding and learning.

A.2 Concepts focusing on Objects of Learning

A.2.1 Stein

Stein [Ste95] reviews concepts of organisational memory and gives recommendations on how to facilitate organisational memory. Stein interprets OM as a process as well as a place to store information. He defines OM is *[...] the means by which knowledge from the past is brought to bear on present activities [...].* He consequently describes types of processes that lead to organisational memories, gives enablers and obstacles and presents approaches of what are the subjects of organisational memory.

In his PhD-Thesis [Ste89], Stein gives a typology of organisational knowledge from a semantic point of view and presents statements on relevant information types:

- techno-scientific knowledge (abstract, descriptive)
- events, people, inputs, outputs (concrete, descriptive)
- policies, values, ethics and strategies (abstract, prescriptive)
- rules, norms, roles and tasks (concrete, prescriptive)

In terms of information representation, Stein suggests to distinguish between three types of information:

- *schema*: represents categorisations of information and are organized hierarchically. It contains individuals' conceptual understanding of the world.

- *script*: describes appropriate sequencing of events in familiar situations. Thus, scripts represent procedural routine knowledge

- *system*: is defined to be a set of inter-related elements which are connected directly or indirectly. Systems offer support for problem-solving and decision-making in new, unfamiliar situations. They can be represented as a (formal or informal) network of organisational structure and dynamics, goals, values and actors.

Stein's review of organisational memory concepts provides a comprehensive overview of the types of information relevant to OM but does hardly give statements on the concrete form and representation of information in these types.

A.2.2 Abecker & van Elst

Abecker & van Elst [AvE01] define an organisational memory to be a means for support of knowledge management by storing data, information and knowledge from different sources within an organisation. They propose a multi-layered architecture, in which information is complemented by meta-data and put into the business context of the organization or an individual. They propose to link information to business process models or to integrate it into a workflow management system to create this context.

In terms of information types, this approach proposes the use of the following components:

- basic information objects (containing declarative, procedural or narrative information)

- structural meta data about information objects (domain-independent, like author)

- domain-specific meta data about information objects (like keywords, classification)

- organisational structure (e.g. departments and roles)

- organisational processes (including the specific involvement of employees)

These information types are organized in ontologies and data sources which are orchestrated in an layered architecture to provide work-context specific information to employees in last consequence.

A.2.3 Eulgem

Eulgem [Eul98] proposes a concept for organisational memories based upon an organisational knowledge model consisting of two layers: *knowledge structure* and *knowledge contents*. Knowledge structure contains organisational concepts (which might be entities, states or activities) and their relationships (both static and dynamic ones, where dynamic ones are represented using process models). Knowledge contents are assigned to a respective concept on the knowledge structure layer and hold the actual information and/or supplementary attributes for the concept.

Summarising, this approach basically uses two types of information:

- organisational processes and how they are embedded within the organisational structure
- information about these organisational aspects

and demands that those two types are intertwined for effective use.

A.2.4 Linger & Burstein

Linger and Burstein [LB98] define an organisational memory to be a dynamic (that is 'changing over time') set of models representing (individual) experiences with and (organisational) views on certain tasks (and how to accomplish them). This models contain links between material used to accomplish a task and the activities, in which these materials are used. Conceptually, they explicitly distinguish individual (specific) models and organisational (general) models, which evolve from individual models in the course of organisational learning.

Summarising, this approach basically uses two types of information:

- tasks which are accomplished through a set of activities
- information which is used in certain activities to accomplish this tasks

The first type is used to represent the models of each involved individual and the (consolidated) organisational model. This distinction is not necessary for the second type, as all models access the same information base and can even be linked using this base.

A.2.5 Ramesh

Ramesh [Ram96] takes a process-oriented point of view to define the necessary contents of an organisational memory. Conceptually, this approach stores models of the involved individuals' views on a certain business process, which are linked by common objects used in this process. The models are made up of tasks, stakeholders (actors), objects (input and output data, like requirements, decisions, etc.) and sources (documents). This approach also allows to attach supplementary, informal information to each of the mentioned concepts.

Summarising, this approach basically used three types of information:

- process models consisting of tasks, actors and required or produced data and their relationships
- documents, in which the required or produced data is represented / stored
- supplementary information to detail tasks, actors, information and documents

A.2.6 Le, Lamontagne & Nguyen

Le, Lamontagne & Nguyen [LLN00] present a multi-dimensional structuring approach for contents of an organisational memory. Their concepts of knowledge units and knowledge networks constitute the types of information considered relevant.

Knowledge units (KU) are the basic building blocks of an OM and are either

- *static* KUs, which contain domain concepts, facts and descriptive information
- *how-to* KUs, which represent task-oriented, procedural knowledge focusing on personal skills and solution strategies

KUs are organized in Knowledge Networks (KN), in which they are intertwined using semantic relationships. A KN can span across different dimensions, each containing KUs and relationships with a certain semantic meaning:

- The *organisational dimension* reflect the organisational structure the OM is embedded in. It provides the actual work context and allows to gain deeper understanding of how the organisation attains its goals
- The *pedagogical dimension* contains KUs that aid understanding of other KUs - thus representing the foundations or prerequisites of the KUs related to the actual organisational context
- The *logical dimension* provides information about logical relationships between KUs and provides support for automated reasoning, which the authors claim could be used to generate creative solutions for problems.

A.2.7 Wargitsch & Wewers

Wargitsch & Webers propose to bring together workflow management systems and organisational memories [WW97]. Consequently, they organize organisational knowledge along work processes and aim at continuous improvement of workflows. As regards contents of the organisational memory, they follow a rather focused, technically oriented approach:

- the basic elements are workflows and their building blocks respectively
- (technical) dependencies between workflows are considered to provide the highest added value in the OM concept
- documents, that contain workflow relevant information (e.g. checklists), are also considered relevant
- communication about workflows is also archived

Appendix B

ISO Topic Map Details

This annex presents the detailed structure of ISO Topic Maps. Is is a complement to chapter 5 - Data Representation Concepts.

B.1 Topics

Topics are the fundamental elements of a topic map. They represent *subjects* of the perceptible world within the topic map. The relationship between *topics* and *subjects* therefore corresponds to the one between *representation* and *referent* in the semiotic thetrahedon (see figure B.1). This relationship is not a direct one but is dependent on an actor's interpretation. Thus a topic map represents one specific view onto reality, which can be shared among individuals by this means.

Figure B.2 shows how topics are connected to other building blocks. Occurrences and Association Roles provide the links to the other central concepts of topic maps and will be described in the respective sections. Topic names and variants are topic interna and will be described in the following section. The concept of subject identifiers and locators as well as reification will be described in the section dealing with advanced functionality of topic maps.

A topic is a symbol used within a topic map to represent one, and only one, subject, in order to allow statements to be made about the subject. A statement is a claim or assertion about a subject (where the subject may be a topic map construct). Topic names, variant names, occurrences, and associations are statements [...] [ISO06a]. This definition of the term *topic* again shows the expressive power of topic maps (as mentioned before): it is possible to give

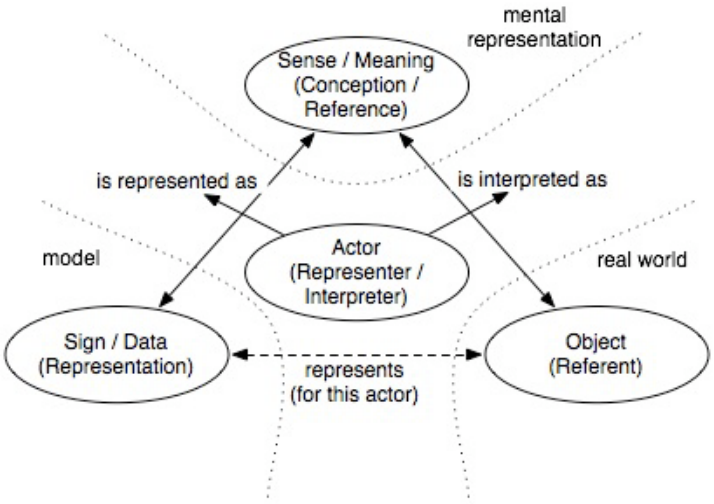

Figure B.1: The Semiotic Thetrahedon (adapted from [FHL+98])

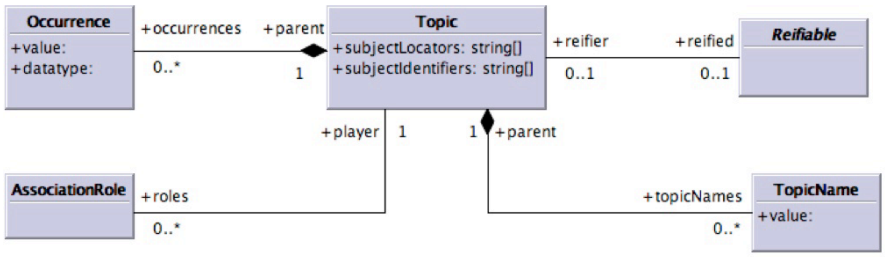

Figure B.2: Topic Maps - Topic (taken from [ISO06a])

statements not only on the semantic relationships between concepts (using associations) but also on the semantics of concepts themselves (using topic names, variants and occurrences).

B.1.1 Topic Names & Variants

Topic elements themselves do not contain any designators but can be referred to using an unique identifier (called *item identifier*). Topics can be extended with topic name elements (one topic may have several topic names and topic names do not have to be unique to a certain topic). A topic name is a textual description of the subject represented by the topic. If a topic has more than one topic names (e.g. for use in different scopes, see later), one name can be defined to be the

default name (see figure B.3).

Variants are alternative forms of a certain topic name. They are used to define synonyms for a given topic name and enable to change the actual form of representation, as they do not necessarily have to be strings (but can e.g. link to encoded audio, see figure B.3). A special form of variants are *sort names*, which enable alphabetical sorting of topics (corresponding to unicode order).

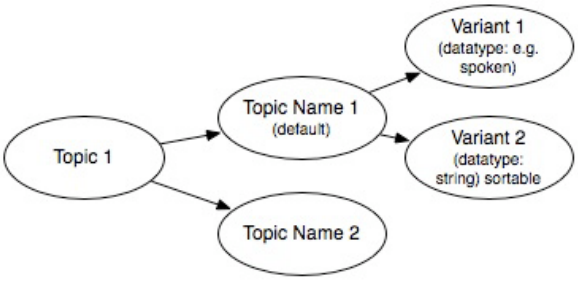

Figure B.3: Topic Maps - Topic Naming

B.2 Associations

Associations represent relationships between subjects and consequently are used to link the topics representing those subjects. They are the second building block necessary to represent a semantic network in a topic map.

An association element itself, like a topic, does not contain an designator. It rather links to a topic that represents its type (see figure B.4). This topic again can contain topic names and variants. Given this information, it is obvious that the drawing in figure 5.2 is inaccurate, as the designator of the association is not drawn as a topic. This is only accounted for better visual representation. Using topics to define the type of an association enables consistent modelling, as multiple associations representing the same kind of relationship are all linked to the same topic representing this association type. This is also an effective means for meta modelling.

Associations allow to define relationships between one or more (arbitrary) subjects (see figure B.4). While the type of the association is defined within the element (as described above), the meaning of the topics attached to the association are defined using *association roles*.

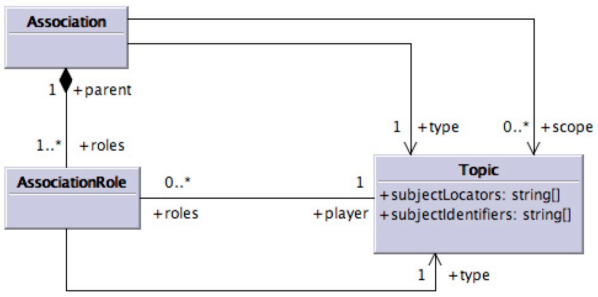

Figure B.4: Topic Maps - Association (taken from [ISO06a])

B.2.1 Association Roles

Every end point of an association has to be defined in an association role. These roles are taken by specific topics (those to be associated). Association roles therefore represent *the involvement of a subject in a relationship represented by an association* [ISO06a].

Like for associations, association roles do not contain any designator but again use topics to define their type. Thus, all the statements given above also apply here.

B.3 Occurrences

Occurrences are links to the 'outer world' (outside the topic map) and complete the expressiveness of the topic map concept by adding index functionality. *An occurrence is a representation of a relationship between a subject and an information resource* [ISO06a]. Occurrences are used to link a topic to information ressources (described textually or using XML inside the topic map or using a generic URI to link to external ressources). Each topic can contain an arbitrary number of occurrences. However, an occurrence cannot be linked to multiple topics.

The definition of an occurrence type is necessary to be able to specify the type of information ressource linked to a topic (or *the nature of the relationship between the subjects and information resources* [ISO06a]),(see figure B.5). This type again is represented by a specific topic (as for associations and association roles, the statements given there also apply here).

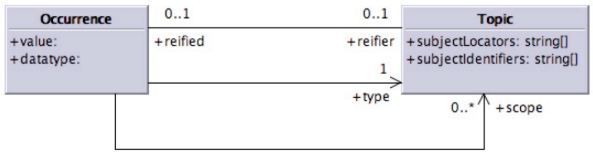

Figure B.5: Topic Maps - Occurrences

B.4 Further Building Blocks - Advanced Functionality

Aside from the three basic building blocks, additional elements extend the expressiveness of topic maps. Figure B.6 gives an overview about the usage of *scopes* and means to model *meta-elements* (or hierarchies). *Subject identifiers and locators* as well as the concept of *reification* are not shown in the figure but are also described in the following.

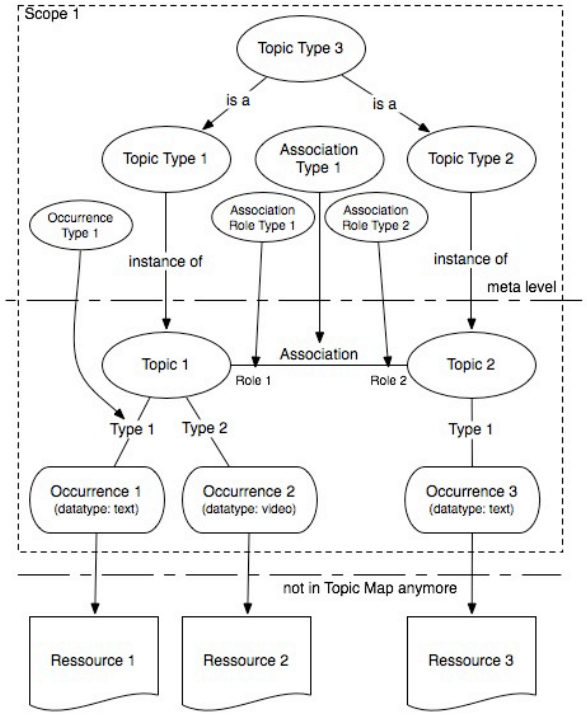

Figure B.6: Topic Maps - Full Overview

B.4.1 Scope

Scopes are a means to define the contexts in which certain statements of a topic map are valid. Scopes are used to describe domains which are represented in a topic map.

Topics themselves actually do not belong to a scope, as only statements (that is, associations, occurrences, topic names and variants) can be valid or invalid (not known to be valid) within a certain context. In fact, the scope is made up of a set of topics. As a topic represents exactly one subject of reality, a scope circumscribes a part of reality (that is, 'through which glasses one is looking onto reality'). A special case of a scope is the *unconstrained scope*, which indicates unlimited validity of the enclosed statements.

In the topic map, the scope itself (i.e. the set of topics defining the scope) is referenced by all topic map elements which are valid in this scope.

B.4.2 Meta-Elements - Types

The topic map standard includes different means of meta modeling (i.e. building a model of a model, defining the types of allowed topics, associations, roles and occurrences). There are three types of meta modeling elements:

Topic Types formally are topics which are connected to other topics using a specific kind of association - a *type-instance relationship* as has be used in figure B.6 for the 'instance of'-associations. This relationship corresponds to that between classes and objects in object-oriented programming and is used to *capture some commonality in a set of subjects* [ISO06a]. Topic types can be considered the central elements of a meta model and define the basic semantic building blocks of a certain modeling domain.

Supertype - Subtype relationships are used whenever a more general type (supertype) and a specialization of that type (subtype) has to be modeled (as has been used in figure B.6 for the 'is a'-associations). This relationship actually is not restricted to use on meta level but can be used on every level of a topic map. It corresponds to inheritance and sub- und super-classes in object-oriented programming. Consequently, 'subtype-supertype'-relationships are transitive, so that a subtype of topic A is also a subtype of topic B, if topic A is a subtype of topic B. For modeling on meta-level, the advantage is, that 'type-instance'-relationships have only to be established between the most specific topic type

and the instance. Taking figure B.7 as an example, 'SR 174 AU' here not only is an instance of 'Golf' but also of 'VW' and of 'Car'.

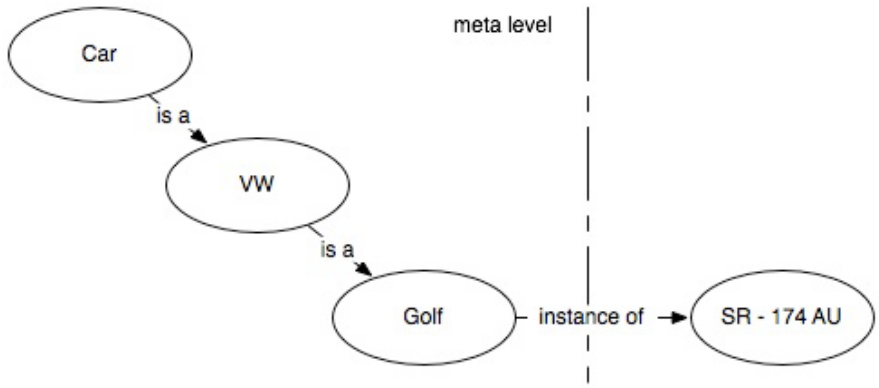

Figure B.7: Topic Maps - Example for super/subtyping and instancing

Association Types, Association Role Types and Occurrence Types are formally represented by the same means - using topic as described above in the sections about associations and occurrences. This mechanism allows to define validity of associations and types of occurrences for a certain topic map. However, it is not possible to define the topic types which can be used with an association type or an occurrences type. This is a major drawback as the meta-model is incomplete and can only be used for partial checking of models. Using associations on meta level to describe the relationships between topic types, association types and occurrence types is a possible workaround. This workaround is fully standard-compliant in terms of formal structure of the map but is not covered by the standard in terms of semantics of the new associations (and consequently needs a meta-meta model for specification).

Other approaches, which examine the feasibility of meta modeling (ontology modeling) in topic maps can be found in the works of Park et al. [PC05], Gulbrandsen [Gul05] and Vatant [Vat04].

B.4.3 Reification

The concept of reification allows to 'attach' a topic (the *reifier*) to any topic map element (other then topics themselves) to include additional information on this

element in the topic map. *For example, creating a topic that represents the relationship represented by an association is reification* [ISO06a].

Using reification, an occurrence can be added to an association. A more complex use case is the assignment of associations to a reifier to put it into relationship with other topics (or reifiers). The whole concepts is recursive, also topic maps themselves are reifiable. In this way, a topic map (reified by a topic) can be part of another topic map.

B.4.4 Merging

The topic map standard defines rules for identifying and algorithms for merging redundant topic map constructs, that is topics and subsequently topic names and variants, occurrences as well as associations and association roles. This is not of immediate relevance for this work and is not described in detail here. For further information refer to [ISO06a].

B.4.5 Subject Identifiers & Locators

Subject identifier and locators are used to formally identify subjects and define their semantics. They are used for precise identification and merging of topic maps. While especially subject identifiers are a highly relevant part of every topic, both are also not of immediate relevance for this work. Thus, only short definitions of the concepts are given here without further explanation:

A subject indicator is an information resource that is referred to from a topic map in an attempt to unambiguously identify the subject represented by a topic to a human being. A subject identifier is a locator that refers to a subject indicator [ISO06a].

A subject locator is a locator that refers to the information resource that is the subject of a topic. The topic thus represents that particular information resource; i.e., the information resource is the subject of the topic [ISO06a].

Both concepts can also be implemented using occurrences but are more special, as they describe the nature a certain topic (and therefore belong to the topic interna). Examples for the use of subject identifiers can be found in part 2 (Data Model), section 7 of the Topic Map Standard [ISO06a].

Appendix C

Topic Map Engine

In this annex, the implementation details of the topic map engine are described. This documentation has been generated automatically from JavaDoc embedded in the source files.

C.1 Package ce.tm4scholion.tm

Package Contents *Page*

Classes

Association 138
TopicMap Engine - Association Associations are the second main element of TopicMaps (beside Topics) and represent the relationships between Topics.

AssociationRole 144
TopicMap Engine - AssociationRole AssociationRoles provide the link between Topics and Assoications.

Manager 147
TopicMap Engine - Manager The Manager-class provides access-routines to manipulate Topic Maps.

Manager.RoleTopic 164
Wrapper-class for management of Association-Role-Topic-Combinations in Sets

Occurrence 165
TopicMap Engine - Occurrence Occurrences are a Topic's links to the 'outer world' (outside the topic map).

Reifiable 170
TopicMap Engine - Reifiable Reifiable TopicMap constructs are constructs which can be further specified or detailed by attaching a topic to them.
Scope 172
TopicMap Engine - Scope Scopes are sets of Topics which span across a set of subjects (context), in which certain Statements are valid.
Statement 176
TopicMap Engine - Statement Statement are no defined construct of the TMDM standard but are described as all constructs of a TopicMap which can have a Scope in which they are valid.
Topic 178
TopicMap Engine - Topic Topics are the central element of each TopicMap and represent subjects of reality.
TopicMap 188
TopicMap Engine - TopicMap TopicMap is the root element of every Topic Map.
TopicMapConstruct 193
TopicMap Engine - TopicMapConstruct The basic element of the TMDM, from which all standardized elements (constructs) are derived.
TopicName 196
TopicMap Engine - TopicName TopicNames are constructs which determine the natural language designator of a Topic (and are actually equal to the name(s) of thr represented subject in most of the cases).

Utils 202
TopicMap Engine - Utils A collection of service routines for internal engine use (not accessible by applications using the engine).
Variant 206
TopicMap Engine - Variant Variants are alternative forms of a certain TopicName.

C.1.1 Class Association

TopicMap Engine - Association Associations are the second main element of TopicMaps (beside Topics) and represent the relationships between Topics. Associations are named using an AssociationType (which is represented using a

Topic) - every Association of the same nature references the same AssociationType. Topics are connected to the Association using AssociationRoles, which describe the role the Topic takes in this Association.

Declaration

public class Association
extends ce.tm4scholion.tm.Statement (in C.1.8, page 176)

Constructor summary

Association() default construtor, generates Association object with UUID as itemIdentifier
Association(String) basic constructor, generates Associaton object with the given String as itemIdentifier

Method summary

addAssociationRole(AssociationRole) add an AssociationRole to this Assciation.
checkTMDMCompliance()
equals(Object)
getAssociatedTopics() get the Topics associated by this Association including the information which AssocationRoles they take
getParent() get the parent element of the Association
getRoles() get the AssociationRoles of this Association (regardless if they are taken or not)
getType() get the AssociationType of the Association
newAssociationRole() request a new AssociationRole for this Association.
removeAssociationRole(AssociationRole) remove an AssociationRole from an Association.
setParent(TopicMap) set the parent element of the Association
setRoles(Set) set the AssociationRoles of this Association
setType(Topic) set the AssociationType of the Association

Constructors

- **Association**
  `public` **Association**`( )`
  - **Description**
    default construtor, generates Association object with UUID as itemIdentifier
- **Association**
  `public` **Association**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates Associaton object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **addAssociationRole**
  `public void` **addAssociationRole**`( AssociationRole` **associationRole** `)`
  - **Description**
    add an AssociationRole to this Assciation. AssociationRoles define the connection points between Topics and Associations
  - **Parameters**
    * `associationRole` -- the AssociationRole to be added to this Association
- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)

check the compliance of this construct (including all directly attached constructs) to the TMDM

- **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant

- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`

- **getAssociatedTopics**
  `public java.util.Map` **getAssociatedTopics**`( )`

  - **Description**

    get the Topics associated by this Association including information on which AssocationRoles they take

  - **Returns** -- a Map of -Tupels, representing which Topics have taken which AssociationRole for this Association

- **getParent**
  `public TopicMap` **getParent**`( )`

  - **Description**

    get the parent element of the Association

  - **Returns** -- the TopicMap the Association is contained in

- **getRoles**
  `public java.util.Set` **getRoles**`( )`

  - **Description**

    get the AssociationRoles of this Association (regardless if they are taken or not)

  - **Returns** -- the AssociationRoles of this Association

- **getType**
  `public Topic` **getType**`( )`

  - **Description**

    get the AssociationType of the Association

  - **Returns** -- the Topic representing the AssocaitonType of the Association

- **newAssociationRole**
  `public AssociationRole` **newAssociationRole**`( )`

  - **Description**

    request a new AssociationRole for this Association. The Assoication-Role is automatically added to the Association

  - **Returns** -- the new AssociationRole

- **removeAssociationRole**
  `public void` **removeAssociationRole**`( AssociationRole` **ar** `)`

  - **Description**

    remove an AssociationRole from an Association. If the role is already taken by a Topic, this Topic is informed of being removed

  - **Parameters**

    * `ar` -- the AssociationRole to be removed from the Association

- **setParent**
  `protected void` **setParent**`( TopicMap` **parent** `)`

  - **Description**

    set the parent element of the Association

  - **Parameters**

    * `parent` -- the TopicMap the Association is contained in

- **setRoles**
  `public void` **setRoles**`( java.util.Set` **roles** `)`

  - **Description**

    set the AssociationRoles of this Association

  - **Parameters**

    * `roles` --

- **setType**
  `protected void` **setType**`( Topic` **type** `)`
  - **Description**
    set the AssociationType of the Association
  - **Parameters**
    * `type` -- the Topic representing the AssocaitonType of the Association

Members inherited from class `ce.tm4scholion.tm.Statement` (in C.1.8, page 176)

- `public Scope` **getScope**`( )`
- `protected` **scope**
- `public void` **setScope**`( Scope` **s** `)`

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.2 Class AssociationRole

TopicMap Engine - AssociationRole AssociationRoles provide the link between Topics and Assoications. They define in which way (role) a Topic particiates in an Association. A particular AssociationRole is always attached to exactly one Association and can be taken by exactly one Topic at the same time. AssocaitionRoles are specified using AssoicationRoleTypes (which are represented using Topics) - every AssociationRole of the same nature references the same AssociationRoleType.

Declaration

public class AssociationRole
extends ce.tm4scholion.tm.Reifiable (in C.1.6, page 170)

Constructor summary

AssociationRole() default construtor, generates AssociationRole object with UUID as itemIdentifier
AssociationRole(String) basic constructor, generates Associaton object with the given String as itemIdentifier

Method summary

checkTMDMCompliance()
equals(Object)
getParent() get the parent element of the AssociationRole
getPlayer() get the Topic which plays the role defined in the AssociationRole
getType() get the AssociationRoleType of the AssociationRole
setParent(Association) set the parent element of the AssociationRole
setPlayer(Topic) set the Topic which plays the role defined in the AssociationRole
setType(Topic) set the AssociationRoleType of the AssociationRole

Constructors

- **AssociationRole**
  `public` **AssociationRole**`( )`
  - **Description**
    default construtor, generates AssociationRole object with UUID as itemIdentifier
- **AssociationRole**
  `public` **AssociationRole**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates Associaton object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant
- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`
- **getParent**
  `public Association` **getParent**`( )`

  - **Description**

    get the parent element of the AssociationRole

  - **Returns** -- the Association the AssociationRole is attached to

- **getPlayer**
  `public Topic` **getPlayer**`( )`

  - **Description**

    get the Topic which plays the role defined in the AssociationRole

  - **Returns** -- the Topic which plays the role defined in the AssociationRole

- **getType**
  `public Topic` **getType**`( )`

  - **Description**

    get the AssociationRoleType of the AssociationRole

  - **Returns** -- the Topic representing the AssocaitonRoleType of the AssociationRole

- **setParent**
  `protected void` **setParent**`( Association` **parent** `)`

  - **Description**

    set the parent element of the AssociationRole

  - **Parameters**

    * `parent` -- the Associaton the AssociationRole is attached to

- **setPlayer**
  `public void` **setPlayer**`( Topic` **player** `)`

  - **Description**

    set the Topic which plays the role defined in the AssociationRole

  - **Parameters**

    * `player` -- the Topic which plays the role defined in the AssociationRole

- **setType**
  `protected void` **setType**`( Topic` **type** `)`
  - **Description**
    set the AssociationRoleType of the AssociationRole
  - **Parameters**
    * `type` -- the Topic representing the AssocaitonRoleType of the AssociationRole

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.3 Class Manager

TopicMap Engine - Manager The Manager-class provides access-routines to manipulate Topic Maps. In general, basic manipulation (that is, altering the TopicMap) should be carried out using the methods defined here, as they ensure consisteny across the whole Topic Map. Actually, the classes for TopicMapConstructs only provide public methods for operations, which cannot corrupt the map's consistency. However, every operation publicly available there can

also be carried out using a (wrapper-)method of the Manager-class. Every Manager-object can handle exactly one TopicMap. In addition to the TopicMap itself, the Manager holds serveral indices of Topics used for typing and of scopes for more efficient and consistent access. Whenever a Manager-object is created, an corresponding TopicMap-object is also set up, which already contains the core topics (for miscellaneous types) defined in the standard. The indices are implemented as maps, where the subjectIdentifiers of the Topics are used as the key values. If a Topic contains several subjectIdentifiers, it is registered in the index several times correspondingly. As subjectIdentifiers have to be locators and thus URIs, the following notation convention for subjectIdentifiers has been defined: "urn:subject:CNxxx", where CN is

- 'a' for Topics representing AssociationTypes
- 'ar' for Topics representing AssociationRoleTypes
- 'o' for Topics representing OccurrenceTypes
- 'tn' for Topics representing TopicNameTypes
- 't' for plain Topics
- 'reifier' for Topics used to reify a reifiable construct

>'xxx' stands for the name of the Topic, which is also used to create the first TopicName of the resprective Topic. Note, that a topic may have several subjectIdentifiers and thus can be used e.g. as a plain Topic and an AssociationType at the same time.

>In addition this notation is also used for the naming of scopes (which actually do not have subjectIdentifiers): 's' for the names of Scopes

Declaration

public class Manager
extends java.lang.Object

Constructor summary

Manager() the default constructor.

Method summary

addOccurrence(Topic, String, String) add an Occurrence of the 'String'-datatype and the given type to a Topic.

addOccurrence(Topic, String, String, String) add an Occurrence of the given datatype and the given type to a Topic.

addSortVariant(TopicName, String) add a sortable Variant of the 'String'-datatype to a TopicName

addTopic(String) add a Topic to the TopicMap using the given name.

addTopic(Topic) add a given Topic to the TopicMap

addTopicName(Topic, String) add a TopicName of the default TopicNameType to a Topic

addTopicName(Topic, String, String) add a TopicName of the given TopicNameType to a Topic.

addTopicName(Topic, String, Topic) add a TopicName of the given TopicNameType to a Topic.

addTopicOfType(String, Topic) add a Topic to the TopicMap using the given name and make it an instance of the given tpye.

addVariant(TopicName, String) add a Variant of the 'String'-datatype to a TopicName

addVariant(TopicName, String, String) add a Variant of the given datatype to a TopicName

associate(String, Set) associate a set of Topics with an Association of the given Type, where the Topics each take the role of the given AssociationRoleType.

containsTopic(Topic) check wether a Topic is contained in a Topic Map

defineAssociationRoleType(String) define a new AssociationRoleType with the given name.

defineAssociationType(String, Set) define a new AssociationType with the given name and the given roles.

defineOccurrenceType(String) define a new OccurrenceType with the given name.

defineScope(Set) define a new Scope and register it to the respective index

defineTopicNameType(String) define a new TopicNameType with the given name.

disassociate(Association) remove an Association from the TopicMap and ensure consistency by removing the respective AssociationRoles from the affected Topics.

generateTopic(String) generates a new Topic using the given name but does not add it to the TopicMap.

getAssociationsParticipatedInRole(Topic, Topic) Retrieves the associations a specified topic participates in in a certain role

getAssociationsRolesBetweenAssociationAndTopic(Topic, Association) Retrieves the association roles that build the bridge between a specified topic and association

getCounterpartTopics(Topic, Topic, Topic) Combines getAssociationsParticipatedInRole and getTopicsInAssocRole and thus retrieves all topics that are assocaited with a given one in a specified role setting

getSubtypes(Topic) Return a Set of Topics which are associated with the given Topic in a superType-subType-Relationship either directly or indirecly (transitivly via other subTypes)

getSupertypes(Topic) Return a Set of Topics which are associated with the given Topic in a superType-subType-Relationship either directly or indirecly (transitivly via other superTypes)

getTopic(String) Retrieve a Topic from the TopicMap based on its subject name

getTopicsInAssocRole(Association, Topic) Retrieves the topics that take a certain role in a specified association

markVariantSortable(Variant) mark a Variant to be 'sortable' (as defined in the TMDM)

nameScope(Scope, String) assign a name to a Scope

reify(Reifiable, Topic) reify a reifiable construct with the given Topic

removeOccurrence(Topic, Occurrence) removes an Occurrence from a Topic

removeTopic(Topic) removes a Topic from the TopicMap.

removeTopicName(Topic, TopicName) removes a TopicName from a Topic
removeVariant(TopicName, Variant) removes a Variant from a TopicName
setScope(Statement, Scope) set a Statement's Scope
setSuperSubType(Topic, Topic) define a Supertype-Subtype-relationship in the terms of the standard
setTypeInstance(Topic, Topic) define a Type-Instance-relationship in the terms of the standard (and implicitly create a TopicType).
topicIsInstanceOf(Topic, Topic) Checks if a Topic is an instance of a given Type.
unReify(Reifiable, Topic) remove a reification

Constructors

- **Manager**
  `public` **Manager**`( )`
  - **Description**
    the default constructor. Creates a Manager object and sets up the topic map and the indices to manage it.

Methods

- **addOccurrence**
  `public Occurrence` **addOccurrence**`( Topic` **t**`, java.lang.String` **value**`, java.lang.String` **type** `)`
  - **Description**
    add an Occurrence of the 'String'-datatype and the given type to a Topic. If the given TopicNameType does not exist, it is created.
  - **Parameters**
    * `t` -- the Topic to which the Occurrence has to be added
    * `value` -- the value of the Occurrence
    * `type` -- the type of the Occurrence
  - **Returns** -- the new Occurrence object

- **addOccurrence**
  `public Occurrence` **addOccurrence**`( Topic` **t**`, java.lang.String` **value**`, java.lang.String` **dataType**`, java.lang.String` **type** `)`
  - **Description**
    add an Occurrence of the given datatype and the given type to a Topic. If the given TopicNameType does not exist, it is created.
  - **Parameters**
    * `t` -- the Topic to which the Occurrence has to be added
    * `value` -- the value of the Occurrence
    * `dataType` -- the dataType of the Occurrence (for default dataTypes, see Utils-class)
    * `type` -- the type of the Occurrence
  - **Returns** -- the new Occurrence object
- **addSortVariant**
  `public Variant` **addSortVariant**`( TopicName` **tn**`, java.lang.String` **variant** `)`
  - **Description**
    add a sortable Variant of the 'String'-datatype to a TopicName
  - **Parameters**
    * `tn` -- the TopicName to which the Variant has to be added
    * `variant` -- the sortable Variant to be added
  - **Returns** -- the new Variant object
- **addTopic**
  `public Topic` **addTopic**`( java.lang.String` **name** `)`
  - **Description**
    add a Topic to the TopicMap using the given name. A respective TopicName item is also created
  - **Parameters**

    * `name` -- the name (subject) of the Topic
  - **Returns** -- the new Topic object

- **addTopic**
  `public Topic` **addTopic**`( Topic` **t** `)`
  - **Description**
    add a given Topic to the TopicMap
  - **Parameters**
    * `t` -- the Topic to be added
  - **Returns** -- the added Topic

- **addTopicName**
  `public TopicName` **addTopicName**`( Topic` **t**`, java.lang.String` **name** `)`
  - **Description**
    add a TopicName of the default TopicNameType to a Topic
  - **Parameters**
    * `t` -- the Topic to which the TopicName has to be added
    * `name` -- the name to be added
  - **Returns** -- the new TopicName object

- **addTopicName**
  `public TopicName` **addTopicName**`( Topic` **t**`, java.lang.String` **name**`, java.lang.String` **type** `)`
  - **Description**
    add a TopicName of the given TopicNameType to a Topic. If the given TopicNameType does not exist, it is created.
  - **Parameters**
    * `t` -- the Topic to which the TopicName has to be added
    * `name` -- the name to be added
    * `type` -- the TopicNameType of the TopicName

  - **Returns** -- the new TopicName object

- **addTopicName**
  `public TopicName` **addTopicName**`( Topic` **t**`, java.lang.String` **name**`, Topic` **type** `)`

  - **Description**

    add a TopicName of the given TopicNameType to a Topic. If the given TopicNameType is not yet registered, it is added to the index.

  - **Parameters**

    * `t` -- the Topic to which the TopicName has to be added
    * `name` -- the name to be added
    * `type` -- the TopicNameType of the TopicName

  - **Returns** -- the new TopicName object

- **addTopicOfType**
  `public Topic` **addTopicOfType**`( java.lang.String` **name**`, Topic` **type** `)`

  - **Description**

    add a Topic to the TopicMap using the given name and make it an instance of the given tpye. A respective TopicName item is also created

  - **Parameters**

    * `name` -- the name (subject) of the Topic
    * `type` -- the Topic Type the new Topic should by an instance of (has be existent)

  - **Returns** -- the new Topic object, null if the given Topic Type does not exist

- **addVariant**
  `public Variant` **addVariant**`( TopicName` **tn**`, java.lang.String` **variant** `)`

  - **Description**

    add a Variant of the 'String'-datatype to a TopicName

- **Parameters**
  - `tn` -- the TopicName to which the Variant has to be added
  - `variant` -- the Variant to be added
- **Returns** -- the new Variant object

- **addVariant**
  `public Variant` **addVariant**`( TopicName` **tn**`, java.lang.String` **variant**`, java.lang.String` **dataType** `)`
  - **Description**
    add a Variant of the given datatype to a TopicName
  - **Parameters**
    - `tn` -- the TopicName to which the Variant has to be added
    - `variant` -- the Variant to be added
    - `dataType` -- the dataType of the Variant (for default dataTypes, see Utils-class)
  - **Returns** -- the new Variant object

- **associate**
  `public Association` **associate**`( java.lang.String` **associationType**`, java.util.Set` **rolesPlayed** `)`
  - **Description**
    associate a set of Topics with an Association of the given Type, where the Topics each take the role of the given AssociationRoleType. If the given AssociationType or an AssociationRoleType does not exist, it is created.
  - **Parameters**
    - `associationType` --
    - `rolesPlayed` --
  - **Returns** --

- **containsTopic**
  `public boolean` **containsTopic**`( Topic` **t** `)`

  - **Description**
    check wether a Topic is contained in a Topic Map
  - **Parameters**
    * `t` -- the Topic to be search
  - **Returns** -- ture if the Topic was found, false otherwise

- **defineAssociationRoleType**
  `public Topic` **defineAssociationRoleType**`( java.lang.String` **name** `)`

  - **Description**
    define a new AssociationRoleType with the given name. The Topic representing the type is registered in the respective index and added to the TopicMap (including a respective TopicName, which type is set to 'urn:subject:tnAssociationRoleType', marking that this is the name of an AssociationRoleType).
  - **Parameters**
    * `name` -- the name of the new AssociationRoleType
  - **Returns** -- the Topic representing the AssociationRoleType

- **defineAssociationType**
  `public Topic` **defineAssociationType**`( java.lang.String` **name**`, java.util.Set` **roles** `)`

  - **Description**
    define a new AssociationType with the given name and the given roles. The Topic representing the associationType and the associationRoleTypes are registered in the respective index and added to the TopicMap (including a respective TopicName, which type is set to 'urn:subject:tnAssocia
    or 'urn:subject:tnAssociationRoleType', marking that this is the name of an AssociationType or an AssociationRoleTypes, respectively). If an AssociationRoleType already exists in the index (because it is already use by another association, it is not created a second time.
  - **Parameters**
    * `name` -- the name of the new AssociationType

    * `roles` -- set of the names of the corresponding AssociationRoleTypes
  - **Returns** -- the Topic representing the AssociationType

- **defineOccurrenceType**
  `public Topic` **defineOccurrenceType**`( java.lang.String` **name** `)`
  - **Description**
    define a new OccurrenceType with the given name. The Topic representing the type is registered in the respective index and added to the TopicMap (including a respective TopicName, which type is set to 'urn:subject:tnOccurrenceType', marking that this is the name of an OccurrenceType).
  - **Parameters**
    * `name` -- the name of the new OccurrenceType
  - **Returns** -- the Topic representing the OccurrenceType

- **defineScope**
  `public Scope` **defineScope**`( java.util.Set` **context** `)`
  - **Description**
    define a new Scope and register it to the respective index
  - **Returns** -- the new Scope object

- **defineTopicNameType**
  `public Topic` **defineTopicNameType**`( java.lang.String` **name** `)`
  - **Description**
    define a new TopicNameType with the given name. The Topic representing the type is registered in the respective index and added to the TopicMap (including a respective TopicName, which type is set to 'urn:subject:tnTopicNameType', marking that this is the name of a TopicNameType).
  - **Parameters**

    * `name` -- the name of the new TopicNameType
  - **Returns** -- the Topic representing the TopicNameType

- **disassociate**
  `public void` **disassociate**`( Association` **a** `)`
  - **Description**

    remove an Association from the TopicMap and ensure consistency by removing the respective AssociationRoles from the affected Topics.
  - **Parameters**
    * `a` -- the Association to be removed

- **generateTopic**
  `public Topic` **generateTopic**`( java.lang.String` **name** `)`
  - **Description**

    generates a new Topic using the given name but does not add it to the TopicMap. A respective TopicName item is also created
  - **Parameters**
    * `name` -- the name (subject) of the Topic
  - **Returns** -- the new Topic object

- **getAssociationsParticipatedInRole**
  `public java.util.Set` **getAssociationsParticipatedInRole**`( Topic` **t**`, Topic` **roleType** `)`
  - **Description**

    Retrieves the associations a specified topic participates in in a certain role
  - **Parameters**
    * `t` -- the topic to be analysed
    * `roleType` -- the assocation role type to be looked for
  - **Returns** -- the associations the given topic participates in in the given role

- **getAssociationsRolesBetweenAssociationAndTopic**
  `public java.util.Set` **getAssociationsRolesBetweenAssociationAndTopic**`( Topic` **t**`, Association` **a** `)`
  - **Description**
    Retrieves the association roles that build the bridge between a specified topic and association
  - **Parameters**
    * `t` -- the topic to be analysed
    * `a` -- the assoication to be analysed
  - **Returns** -- the association roles that build the bridge between the given topic and association

- **getCounterpartTopics**
  `public java.util.Set` **getCounterpartTopics**`( Topic` **givenTopic**`, Topic` **givenRoleType**`, Topic` **searchedRoleType** `)`
  - **Description**
    Combines getAssociationsParticipatedInRole and getTopicsInAssocRole and thus retrieves all topics that are assocaited with a given one in a specified role setting
  - **Parameters**
    * `givenTopic` -- the originating topic (to be analysed)
    * `givenRoleType` -- the role type of the given topic to be looked for
    * `searchedRoleType` -- the role type of the counterpart topics to be looked for
  - **Returns** -- all topics that are associated with the given topic in a certain - given - role setting

- **getSubtypes**
  `public java.util.Set` **getSubtypes**`( Topic` **t** `)`
  - **Description**

Return a Set of Topics which are associated with the given Topic in a superType-subType-Relationship either directly or indirecly (transitivly via other subTypes)

- **Parameters**
  - `t` -- the Topic to be evaluated
- **Returns** -- a Set of Topics which are subTypes of the given Topic

- **getSupertypes**
  `public java.util.Set` **getSupertypes**`( Topic` **t** `)`
  - **Description**
    Return a Set of Topics which are associated with the given Topic in a superType-subType-Relationship either directly or indirecly (transitivly via other superTypes)
  - **Parameters**
    - `t` -- the Topic to be evaluated
  - **Returns** -- a Set of Topics which are superTypes of the given Topic

- **getTopic**
  `public Topic` **getTopic**`( java.lang.String` **name** `)`
  - **Description**
    Retrieve a Topic from the TopicMap based on its subject name
  - **Parameters**
    - `name` -- the subject name to search for
  - **Returns** -- the fond topic, null, if no topic was found

- **getTopicsInAssocRole**
  `public java.util.Set` **getTopicsInAssocRole**`( Association` **a**`, Topic` **roleType** `)`
  - **Description**
    Retrieves the topics that take a certain role in a specified association
  - **Parameters**

    * `a` -- the assoication to be analysed
    * `roleType` -- the assocation role type to be looked for
  - **Returns** -- the topics that take the given role in the given associaiton

- **markVariantSortable**
  `public void` **markVariantSortable**`( Variant` **v** `)`
  - **Description**
    mark a Variant to be 'sortable' (as defined in the TMDM)
  - **Parameters**
    * `v` -- the Variant to be marked as 'sortable'

- **nameScope**
  `public void` **nameScope**`( Scope` **s**`, java.lang.String` **name** `)`
  - **Description**
    assign a name to a Scope
  - **Parameters**
    * `s` -- the Scope to be named
    * `name` -- the name to be assigned to the Scope

- **reify**
  `public void` **reify**`( Reifiable` **reified**`, Topic` **reifier** `)`
  - **Description**
    reify a reifiable construct with the given Topic
  - **Parameters**
    * `reified` -- the construct to be reified
    * `reifier` -- the Topic which is used as the reifier

- **removeOccurrence**
  `public void` **removeOccurrence**`( Topic` **t**`, Occurrence` **o** `)`
  - **Description**
    removes an Occurrence from a Topic

  - **Parameters**
    * `t` -- the Occurrence from which the TopicName has to be removed
    * `tn` -- the Occurrence to be removed

- **removeTopic**
  `public boolean` **removeTopic**`( Topic` **t** `)`
  - **Description**
    removes a Topic from the TopicMap. The Topic is not removed, if it is used to represent a type (except for topicTypes). If it represents a topicType, all corresponding typeInstance-Associations are removed before the Topic is deleted from the TopicMap and the topicTypes-index.
  - **Parameters**
    * `t` -- the Topic to be removed
  - **Returns** -- true if the Topic has been successfully removed, false otherwise

- **removeTopicName**
  `public void` **removeTopicName**`( Topic` **t**`, TopicName` **tn** `)`
  - **Description**
    removes a TopicName from a Topic
  - **Parameters**
    * `t` -- the Topic from which the TopicName has to be removed
    * `tn` -- the TopicName to be removed

- **removeVariant**
  `public void` **removeVariant**`( TopicName` **tn**`, Variant` **v** `)`
  - **Description**
    removes a Variant from a TopicName
  - **Parameters**
    * `tn` -- the TopicName from which the Variant has to be removed
    * `v` -- the Variant to be removed

- **setScope**
  `public void` **setScope**`( Statement` **statement**`, Scope` **scope** `)`
  - **Description**
    set a Statement's Scope
  - **Parameters**
    * `statement` -- the Statement to be added to the scope
    * `scope` -- the Sope to which the Statement has to be added
- **setSuperSubType**
  `public void` **setSuperSubType**`( Topic` **superType**`, Topic` **subType** `)`
  - **Description**
    define a Supertype-Subtype-relationship in the terms of the standard
  - **Parameters**
    * `superType` -- the Topic which is the supertype
    * `subType` -- the Topic which is the subtype
- **setTypeInstance**
  `public void` **setTypeInstance**`( Topic` **instance**`, Topic` **type** `)`
  - **Description**
    define a Type-Instance-relationship in the terms of the standard (and implicitly create a TopicType). If the Topic given as 'type' is not yet registered as a topicType, it is add to the respective index.
  - **Parameters**
    * `instance` -- the Topic which is the instance
    * `type` -- the Topic to be used as the topicType
- **topicIsInstanceOf**
  `public boolean` **topicIsInstanceOf**`( Topic` **t**`, Topic` **type** `)`

  - **Description**

    Checks if a Topic is an instance of a given Type. Not only evaluates direct type-instance-relationships but also checks supertypes of the given topic's types (necessary because of transitive characteristics of this relationship).

  - **Parameters**

    * `t` -- the Topic to be evaluated
    * `type` -- the Type to be checked for

  - **Returns** -- true if the given Topic is an instance of the given Topic-Type, false otherwise

- **unReify**
  `public void` **unReify**`( Reifiable` **reified**`, Topic` **reifier** `)`

  - **Description**

    remove a reification

  - **Parameters**

    * `reified` -- the construct, from which the reification has to be removed
    * `reifier` -- the Topic, from which the reification has to be removed

C.1.4 Class Manager.RoleTopic

Wrapper-class for management of Association-Role-Topic-Combinations in Sets

Declaration

public static class Manager.RoleTopic
extends java.lang.Object

Field summary

role
topic

Constructor summary

Manager.RoleTopic(String, Topic)

Fields

- public java.lang.String **role**
- public Topic **topic**

Constructors

- **Manager.RoleTopic**
  `public` **Manager.RoleTopic**`( java.lang.String` **role**`, Topic` **topic** `)`

C.1.5 Class Occurrence

TopicMap Engine - Occurrence Occurrences are a Topic's links to the 'outer world' (outside the topic map). An occurrence is a representation of a relationship between a subject and an information resource. Occurrences always have an OccurrenceType (which is represented using a Topic) - every Occurrence of the same nature references the same OccurrenceType. Every Occurrence is attached to exactly one Topic.

Declaration

public class Occurrence
extends ce.tm4scholion.tm.Statement (in C.1.8, page 176)

Field summary

dataType
value

Constructor summary

Occurrence() default construtor, generates Occurrence object with UUID as itemIdentifier
Occurrence(String) basic constructor, generates Occurrence object with the given String as itemIdentifier

Method summary

checkTMDMCompliance()
equals(Object)
getDataType() get the dataType of the Occurrence
getParent() set the parent element of the Occurrence
getType() get the OccurrenceType of the Occurrence
getValue() get the value of the Occurrence (contains the information ressource or the link to it, respectively)
setDataType(String) set the dataType of the Occurrence
setParent(Topic) get the parent element of the Occurrence
setType(Topic) set the OccurrenceType of the Occurrence
setValue(String) set the value of the Occurrence (contains the information ressource or the link to it, respectively)

Fields

- protected java.lang.String **value**
- protected java.lang.String **dataType**

Constructors

- **Occurrence**
  `public` **Occurrence**`( )`
  - **Description**
    default construtor, generates Occurrence object with UUID as itemIdentifier

- **Occurrence**
  `public` **Occurrence**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates Occurrence object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant
- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`
- **getDataType**
  `public java.lang.String` **getDataType**`( )`
  - **Description**
    get the dataType of the Occurrence
  - **Returns** -- the dataType of the Occurrence
- **getParent**
  `public Topic` **getParent**`( )`
  - **Description**
    set the parent element of the Occurrence

  - **Parameters**
    * `parent` -- the Topic the Occurrence is bound to

- **getType**
  `public Topic` **getType**`( )`
  - **Description**
    get the OccurrenceType of the Occurrence
  - **Returns** -- the Topic representing the OccurrenceType of the Occurrence

- **getValue**
  `public java.lang.String` **getValue**`( )`
  - **Description**
    get the value of the Occurrence (contains the information ressource or the link to it, respectively)
  - **Returns** -- the value of the Occurrence

- **setDataType**
  `protected void` **setDataType**`( java.lang.String` **dataType** `)`
  - **Description**
    set the dataType of the Occurrence
  - **Parameters**
    * `dataType` -- the dataType of the Occurrence

- **setParent**
  `protected void` **setParent**`( Topic` **parent** `)`
  - **Description**
    get the parent element of the Occurrence
  - **Returns** -- the Topic the Occurrence is bound to

- **setType**
  `protected void` **setType**`( Topic` **type** `)`

- **Description**

  set the OccurrenceType of the Occurrence

- **Parameters**

  * `type` -- the Topic representing the OccurrenceType of the Occurrence

- **setValue**
  `protected void` **setValue**`( java.lang.String` **value** `)`

  - **Description**

    set the value of the Occurrence (contains the information ressource or the link to it, respectively)

  - **Parameters**

    * `value` -- the value of the Occurrence

Members inherited from class `ce.tm4scholion.tm.Statement` (in C.1.8, page 176)

- `public Scope` **getScope**`( )`
- `protected` **scope**
- `public void` **setScope**`( Scope` **s** `)`

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**

- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.6 Class Reifiable

TopicMap Engine - Reifiable TopicMap constructs are constructs which can be further specified or detailed by attaching a topic to them. With reification, it is for example possible to add an occurrence to an association or to assign a name to an occurrence. A more complex use case is be the assignment of associations to a reifier to put it into relationship with other topics (or reifiers). Reifiable constructs are all standardized constructs except Topics themselves.

Declaration

public abstract class Reifiable
extends ce.tm4scholion.tm.TopicMapConstruct (in C.1.11, page 193)

All known subclasses

Variant (in C.1.14, page 206), TopicName (in C.1.12, page 196), TopicMap (in C.1.10, page 188), Statement (in C.1.8, page 176), Occurrence (in C.1.5, page 165), AssociationRole (in C.1.2, page 144), Association (in C.1.1, page 138)

Constructor summary

Reifiable(String) basic constructor

Method summary

getReifier() get the reifing Topic of the reifiable construct. null if there is no reifier
setReifier(Topic) set the reifing Topic of the reifiable construct

Constructors

- **Reifiable**
  `public` **Reifiable**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **getReifier**
  `public Topic` **getReifier**`( )`
  - **Description**
    get the reifing Topic of the reifiable construct. null if there is no reifier
  - **Returns** -- the reifing Topic
- **setReifier**
  `public void` **setReifier**`( Topic` **reifier** `)`
  - **Description**
    set the reifing Topic of the reifiable construct
  - **Parameters**
    * `reifier` -- the Topic reifing the reifiable construct

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**

- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.7 Class Scope

TopicMap Engine - Scope Scopes are sets of Topics which span across a set of subjects (context), in which certain Statements are valid. Scopes are no defined construct of the TMDM standard but are used to describe the validity of Statements and have been introduced as a class for convenience. Scopes by definition do not have a designator. For reasons of adressablity, this implementation allows to define names for scopes, which are represented in a TMDM-compliant way using a Topic of a specific type (ScopeNameTopicType).

Declaration

public class Scope
extends java.lang.Object

Constructor summary

Scope() default constructor, used to construct a new Scope

Method summary

addStatement(Statement) add a Statement to the Scope, thus defining it to be valid within the Scope
addToContext(Topic) add a Topic to the Scope, thus defining/extending the context in which Statements contained in this Scope are valid
equals(Object)
getContents() get the set of Statements which are valid in this Scope
getContext() get the set of Topics determining the Scope's validity context
getId() get the ID of the Scope
getName() get the name of the Scope

getScopeNames() gets the names assigned to this Scope in a Set of Strings
removeFromContext(Topic) remove a Topic from the Scope, thus restricting the context in which Statements contained in this Scope are valid
removeStatement(Statement) remove a Statement from the Scope, thus defining it not be known to be valid anymore within the Scope
setContents(Set) set the set of Statements in this Scope
setContext(Set) set the Topics defining the Scope's validity context as a whole
setId(String) set the ID of the Scope
setName(Topic) set the Topic containing the designator of this Scope

Constructors

- **Scope**
  `public` **Scope**`( )`
  - **Description**
    default constructor, used to construct a new Scope

Methods

- **addStatement**
  `protected void` **addStatement**`( Statement` **s** `)`
  - **Description**
    add a Statement to the Scope, thus defining it to be valid within the Scope
  - **Parameters**
    * `s` -- the Statement to be added
- **addToContext**
  `public void` **addToContext**`( Topic` **t** `)`
  - **Description**

add a Topic to the Scope, thus defining/extending the context in which Statements contained in this Scope are valid

- **Parameters**
  - `t` -- the Topic to be added

- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`

- **getContents**
  `public java.util.Set` **getContents**`( )`
  - **Description**
    get the set of Statements which are valid in this Scope
  - **Returns** -- the set of Statements which are valid in this Scope

- **getContext**
  `public java.util.Set` **getContext**`( )`
  - **Description**
    get the set of Topics determining the Scope's validity context
  - **Returns** -- the set of Topics determining the Scope's validity context

- **getId**
  `public java.lang.String` **getId**`( )`
  - **Description**
    get the ID of the Scope
  - **Returns** --

- **getName**
  `public Topic` **getName**`( )`
  - **Description**
    get the name of the Scope
  - **Returns** -- the name of the Scope

- **getScopeNames**
  `public java.util.Set` **getScopeNames**`( )`

  - **Description**

    gets the names assigned to this Scope in a Set of Strings

  - **Returns** -- the names assigned to this Scope

- **removeFromContext**
  `public void` **removeFromContext**`( Topic` **t** `)`

  - **Description**

    remove a Topic from the Scope, thus restricting the context in which Statements contained in this Scope are valid

  - **Parameters**

    * `t` -- the Topic to be removed

- **removeStatement**
  `protected void` **removeStatement**`( Statement` **s** `)`

  - **Description**

    remove a Statement from the Scope, thus defining it not be known to be valid anymore within the Scope

  - **Parameters**

    * `s` -- the Statement to be removed

- **setContents**
  `public void` **setContents**`( java.util.Set` **contents** `)`

  - **Description**

    set the set of Statements in this Scope

  - **Parameters**

    * `contents` --

- **setContext**
  `public void` **setContext**`( java.util.Set` **context** `)`

  - **Description**

    set the Topics defining the Scope's validity context as a whole

  - **Parameters**

    * `context` -- the set of Topics which determine the Scope's validitiy context

- **setId**
  `public void` **setId**`( java.lang.String` **id** `)`
  - **Description**
    set the ID of the Scope
  - **Parameters**
    * `id` --

- **setName**
  `protected void` **setName**`( Topic` **t** `)`
  - **Description**
    set the Topic containing the designator of this Scope
  - **Parameters**
    * `t` -- the Topic containing the designator of this Scope

C.1.8 Class Statement

TopicMap Engine - Statement Statement are no defined construct of the TMDM standard but are described as all constructs of a TopicMap which can have a Scope in which they are valid. Following the TMDM, Statements are Assocations, AssociationRoles, Occurrences, TopicNames, TopicNames and Variants.

Declaration

public abstract class Statement
extends ce.tm4scholion.tm.Reifiable (in C.1.6, page 170)

All known subclasses

Variant (in C.1.14, page 206), TopicName (in C.1.12, page 196), Occurrence (in C.1.5, page 165), Association (in C.1.1, page 138)

Field summary

scope

Constructor summary

Statement(String) basic constructor

Method summary

getScope() get the Scope of the Statement
setScope(Scope) set the Scope for the Statement

Fields

- protected Scope **scope**

Constructors

- **Statement**
  `public` **Statement**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **getScope**
  `public Scope` **getScope**`( )`
  - **Description**
    get the Scope of the Statement
  - **Returns** -- the Scope of the Statement
- **setScope**
  `public void` **setScope**`( Scope` **s** `)`

- **Description**
  set the Scope for the Statement
- **Parameters**
  - `s` -- the Scope to be set for this Statement

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.9 Class Topic

TopicMap Engine - Topic Topics are the central element of each TopicMap and represent subjects of reality. Topics have subjectIdentifiers, which are locators (i.e. URIs) that link to resources that describe the nature of the topic. SubjectLocators link to a resource which is the subject of the topic itself. A topic might have occurrences, which are representations of a relationships between a subject and an information resource. Topics might also have a TopicType, which describes the type of the subject represented by teh topic. The TopicType, however, is not stored directly within the Topic but is represented using a predefined form of Assocaiton (TypeInstance-Association). Topics are designated using TopicNames, i.e. a single Topic can have several names. A Topic may take roles in

Associations to be linked to other Topics. Topics can also be used to reify other constructs (only reifiable ones), that is to further specify and detail constructs by the means of attaching a Topic.

Declaration

public class Topic
extends ce.tm4scholion.tm.TopicMapConstruct (in C.1.11, page 193)

Field summary

subjectIdentifiers
subjectLocators

Constructor summary

Topic() default construtor, generates Topic object with UUID as itemIdentifier
Topic(String) basic constructor, generates Topic object with the given String as itemIdentifier

Method summary

addOccurrence(Occurrence) adds an Occurrence to the Topic
addRolePlayed(AssociationRole) add an AssocationRole this Topic plays (invoked by AssociationRole to establish backlink)
addSubjectIdentifier(String) add a new subjectIdentifier to the Topic, only performed if the subjectIdentifier is a locator in terms of the TMDM
addSubjectLocator(String) add a new subjectLocator to the Topic, only performed if the subjectLocator is a locator in terms of the TMDM
addTopicName(TopicName) add a TopicName to the Topic
checkTMDMCompliance()
equals(Object)

getAssociatedAssociations() get the Associations associated by this Topic including information on which Topic is involved via which AssocationRole in an Association
getOccurrences() get the set of Occurrences of the Topic
getParent() get the parent element of the Topic
getReified() get the construct this Topic reifies
getRolesPlayed() get the set of roles played by this topic
getSubjectIdentifiers() get the set of subjectIdentifier of the Topic
getSubjectLocators() get the set of subjectLocators of the Topic
getTopicNames() get the set of TopicNames of this Topic
newOccurrence(String, String) generate a new Occurrence for the Topic with the given value and datatype.
newTopicName(String) generate a new TopicName for the Topic with the given designator.
removeOccurrence(Occurrence) remove an Occurrence from the Topic
removeRolePlayed(AssociationRole) remove a played role from this Topic (invoked by AssociationRole to remove backlink)
removeSubjectIdentifier(String) remove a subjectIdentifier from the Topic
removeSubjectLocator(String) remove a subjectLocator from the Topic
removeTopicName(TopicName) remove a TopicName from the Topic
setOccurrences(Set) set the set of Occurrences of the Topic
setParent(TopicMap) set the parent element of the Topic
setReified(Reifiable) set the reified construct (invoked by Reifiable to establish backlink)
setRolesPlayed(Set) set the set of roles played by this topic
setSubjectIdentifiers(Set) set the set of subjectIdentifiers of the Topic
setSubjectLocators(Set) set the set of subjectLocators of the Topic
setTopicNames(Set) set the set of TopicNames of this Topic

Fields

- protected java.util.Set **subjectLocators**
- protected java.util.Set **subjectIdentifiers**

Constructors

- **Topic**
  `public` **Topic**`( )`
  - **Description**
    default construtor, generates Topic object with UUID as itemIdentifier
- **Topic**
  `public` **Topic**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates Topic object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **addOccurrence**
  `public void` **addOccurrence**`( Occurrence` **occurrence** `)`
  - **Description**
    adds an Occurrence to the Topic
  - **Parameters**
    * `occurrence` -- the Occurrence object to be added
- **addRolePlayed**
  `protected void` **addRolePlayed**`( AssociationRole` **associationRole** `)`

  - **Description**

    add an AssocationRole this Topic plays (invoked by AssociationRole to establish backlink)

  - **Parameters**

    * `associationRole` -- the role to be played by the topic

- **addSubjectIdentifier**
  `public boolean` **addSubjectIdentifier**`( java.lang.String` **subjectIdentifier** `)`

  - **Description**

    add a new subjectIdentifier to the Topic, only performed if the subjectIdentifier is a locator in terms of the TMDM

  - **Parameters**

    * `subjectIdentifier` -- the subjectLocator to be added

  - **Returns** -- true, if the subjectIdentifier was successfully added, false otherwise

- **addSubjectLocator**
  `public boolean` **addSubjectLocator**`( java.lang.String` **subjectLocator** `)`

  - **Description**

    add a new subjectLocator to the Topic, only performed if the subjectLocator is a locator in terms of the TMDM

  - **Parameters**

    * `subjectLocator` -- the subjectLocator to be added

  - **Returns** -- true, if the subjectLocator was successfully added, false otherwise

- **addTopicName**
  `public void` **addTopicName**`( TopicName` **topicName** `)`

  - **Description**

    add a TopicName to the Topic

- **Parameters**
  - `topicName` -- the TopicName object to be added

- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`

  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant

- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`

- **getAssociatedAssociations**
  `public java.util.Map` **getAssociatedAssociations**`( )`

  - **Description**
    get the Associations associated by this Topic including information on which Topic is involved via which AssocationRole in an Association
  - **Returns** -- a Map of -Tupels, representing which Associations are associated to the Topic via which AssocationRole

- **getOccurrences**
  `public java.util.Set` **getOccurrences**`( )`

  - **Description**
    get the set of Occurrences of the Topic
  - **Returns** -- the set of Occurrences of the Topic

- **getParent**
  `public TopicMap` **getParent**`( )`

  - **Description**
    get the parent element of the Topic

  - **Returns** -- the TopicMap the Topic is contained in

- **getReified**
  `public Reifiable` **getReified**`( )`

  - **Description**
    get the construct this Topic reifies
  - **Returns** -- the construct which is reified by this Topic, null, if the Topic is no reifier

- **getRolesPlayed**
  `public java.util.Set` **getRolesPlayed**`( )`

  - **Description**
    get the set of roles played by this topic
  - **Returns** -- the set of roles played by this Topic

- **getSubjectIdentifiers**
  `public java.util.Set` **getSubjectIdentifiers**`( )`

  - **Description**
    get the set of subjectIdentifier of the Topic
  - **Returns** -- the set of subjectIdentifier of the Topic

- **getSubjectLocators**
  `public java.util.Set` **getSubjectLocators**`( )`

  - **Description**
    get the set of subjectLocators of the Topic
  - **Returns** -- the set of subjectLocators of the Topic

- **getTopicNames**
  `public java.util.Set` **getTopicNames**`( )`

  - **Description**
    get the set of TopicNames of this Topic
  - **Returns** -- the set of TopicNames of this Topic

- **newOccurrence**
  `public Occurrence` **newOccurrence**`( java.lang.String` **value**`, java.lang.String` **dataType** `)`
  - **Description**
    generate a new Occurrence for the Topic with the given value and datatype.
  - **Parameters**
    * `value` -- the value of the new Occurrence
    * `dataType` -- the datatype of the new Occurrence
  - **Returns** -- the new Occurrence object
- **newTopicName**
  `public TopicName` **newTopicName**`( java.lang.String` **value** `)`
  - **Description**
    generate a new TopicName for the Topic with the given designator.
  - **Parameters**
    * `value` -- the designator to be used for the new TopicName
  - **Returns** -- the new TopicName object
- **removeOccurrence**
  `public void` **removeOccurrence**`( Occurrence` **o** `)`
  - **Description**
    remove an Occurrence from the Topic
  - **Parameters**
    * `o` -- the Occurrence to be removed
- **removeRolePlayed**
  `protected void` **removeRolePlayed**`( AssociationRole` **ar** `)`
  - **Description**
    remove a played role from this Topic (invoked by AssociationRole to remove backlink)

  - **Parameters**
    * `ar` -- the role to be removed

- **removeSubjectIdentifier**
  `protected void` **removeSubjectIdentifier**`( java.lang.String` **subjectIdentifier** `)`
  - **Description**
    remove a subjectIdentifier from the Topic
  - **Parameters**
    * `subjectIdentifier` -- the subjectLocator to be removed

- **removeSubjectLocator**
  `public void` **removeSubjectLocator**`( java.lang.String` **subjectLocator** `)`
  - **Description**
    remove a subjectLocator from the Topic
  - **Parameters**
    * `subjectLocator` -- the subjectLocator to be removed

- **removeTopicName**
  `public void` **removeTopicName**`( TopicName` **tn** `)`
  - **Description**
    remove a TopicName from the Topic
  - **Parameters**
    * `tn` -- the TopicName to be removed

- **setOccurrences**
  `public void` **setOccurrences**`( java.util.Set` **occurrences** `)`
  - **Description**
    set the set of Occurrences of the Topic
  - **Parameters**

`* occurrences` --

- **setParent**
  `protected void` **setParent**`( TopicMap` **parent** `)`
  - **Description**
    set the parent element of the Topic
  - **Parameters**
    `* parent` -- the TopicMap the Topic is contained in
- **setReified**
  `public void` **setReified**`( Reifiable` **reified** `)`
  - **Description**
    set the reified construct (invoked by Reifiable to establish backlink)
  - **Parameters**
    `* reified` -- the construct to be reified
- **setRolesPlayed**
  `public void` **setRolesPlayed**`( java.util.Set` **rolesPlayed** `)`
  - **Description**
    set the set of roles played by this topic
  - **Parameters**
    `* rolesPlayed` --
- **setSubjectIdentifiers**
  `public void` **setSubjectIdentifiers**`( java.util.Set` **subjectIdentifiers** `)`
  - **Description**
    set the set of subjectIdentifiers of the Topic
  - **Parameters**
    `* subjectIdentifiers` --

- **setSubjectLocators**
  `public void` **setSubjectLocators**`( java.util.Set` **subjectLocators** `)`
  - **Description**
    set the set of subjectLocators of the Topic
  - **Parameters**
    * `subjectLocators` --

- **setTopicNames**
  `public void` **setTopicNames**`( java.util.Set` **topicNames** `)`
  - **Description**
    set the set of TopicNames of this Topic
  - **Parameters**
    * `topicNames` --

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.10 Class TopicMap

TopicMap Engine - TopicMap TopicMap is the root element of every Topic Map. It contains references to all topics and associations which are part of the Topic Map.

Declaration

public class TopicMap
extends ce.tm4scholion.tm.Reifiable (in C.1.6, page 170)

Constructor summary

TopicMap() default construtor, generates TopicMap object with UUID as itemIdentifier
TopicMap(String) basic constructor, generates TopicMap object with the given String as itemIdentifier

Method summary

addAssociation(Association) add an Association to the TopicMap
addTopic(Topic) add a Topic to this TopicMap
checkTMDMCompliance()
getAssociations() get the set of Associations of the TopicMap
getTopics() get the set of Topics of the TopicMap
newAssociation() generate a new Association in the TopicMap with an UUID-itemIdentifier.
newTopic() generate a new Topic in the TopicMap with an UUID-itemIdentifier.
removeAssociation(Association) remove a Association from the TopicMap (without checking any dependencies, this is done in the resprective Manager-routine)
removeTopic(Topic) remove a Topic from the TopicMap (without checking any dependencies, this is done in the resprective Manager-routine)
setAssociations(Set) set the Associations of the TopicMap as a whole
setTopics(Set) set the Topics of the TopicMap as a whole

Constructors

- **TopicMap**
  `public` **TopicMap**`( )`

  - **Description**

    default construtor, generates TopicMap object with UUID as itemIdentifier

- **TopicMap**
  `public` **TopicMap**`( java.lang.String` **itemIdentifier** `)`

  - **Description**

    basic constructor, generates TopicMap object with the given String as itemIdentifier

  - **Parameters**

    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **addAssociation**
  `protected void` **addAssociation**`( Association` **association** `)`

  - **Description**

    add an Association to the TopicMap

  - **Parameters**

    * `association` -- the Association to be added

- **addTopic**
  `protected void` **addTopic**`( Topic` **topic** `)`

  - **Description**

    add a Topic to this TopicMap

  - **Parameters**

    * `topic` -- the Topic to be added

- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`

  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant

- **getAssociations**
  `public java.util.Set` **getAssociations**`( )`
  - **Description**
    get the set of Associations of the TopicMap
  - **Returns** -- the set of Associations of the TopicMap

- **getTopics**
  `public java.util.Set` **getTopics**`( )`
  - **Description**
    get the set of Topics of the TopicMap
  - **Returns** -- the set of Topics of the TopicMap

- **newAssociation**
  `protected Association` **newAssociation**`( )`
  - **Description**
    generate a new Association in the TopicMap with an UUID-itemIdentifier.
  - **Returns** -- the new Association object

- **newTopic**
  `protected Topic` **newTopic**`( )`
  - **Description**
    generate a new Topic in the TopicMap with an UUID-itemIdentifier.
  - **Returns** -- the new Topic object

- **removeAssociation**
  `protected void` **removeAssociation**`( Association` **a** `)`

  - **Description**
    remove a Association from the TopicMap (without checking any dependencies, this is done in the resprective Manager-routine)
  - **Parameters**
    * `t` -- the Association to be removed

- **removeTopic**
  `protected void` **removeTopic**`( Topic` **t** `)`
  - **Description**
    remove a Topic from the TopicMap (without checking any dependencies, this is done in the resprective Manager-routine)
  - **Parameters**
    * `t` -- the Topic to be removed

- **setAssociations**
  `protected void` **setAssociations**`( java.util.Set` **associations** `)`
  - **Description**
    set the Associations of the TopicMap as a whole
  - **Parameters**
    * `associations` -- the Associations to be set as the contents of the TopicMap

- **setTopics**
  `protected void` **setTopics**`( java.util.Set` **topics** `)`
  - **Description**
    set the Topics of the TopicMap as a whole
  - **Parameters**
    * `topics` -- the Topics to be set as the contents of the TopicMap

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.11 Class TopicMapConstruct

TopicMap Engine - TopicMapConstruct The basic element of the TMDM, from which all standardized elements (constructs) are derived. Basically only defines the Set of itemIdentifiers every construct has to have. ItemIdentifiers have to be locators (i.e. URIs) and have to be unique for the TopicMap the Construct is part of.

Declaration

public abstract class TopicMapConstruct
extends java.lang.Object

All known subclasses

Variant (in C.1.14, page 206), TopicName (in C.1.12, page 196), TopicMap (in C.1.10, page 188), Topic (in C.1.9, page 178), Statement (in C.1.8, page 176), Reifiable (in C.1.6, page 170), Occurrence (in C.1.5, page 165), AssociationRole (in C.1.2, page 144), Association (in C.1.1, page 138)

Field summary

firstItemIdentifier
itemIdentifiers

Constructor summary

TopicMapConstruct(String) basic constructor, is invoked by subtypes to generate a respective object with the given String as itemIdentifier.

Method summary

addItemIdentifier(String) add an itemIdentifier to a construct. itemIdentifiers have to locators (that is, URIs) and be unique for the TopicMap the construct is part of

checkTMDMCompliance() check the compliance of this construct (including all directly attached constructs) to the TMDM

getFirstItemIdentifier() get the first itemIdentifier of a TopicMapConstruct

getItemIdentifiers() get the itemIdentifiers of the construct

setFirstItemIdentifier(String) set the first itemIdentifier of a TopicMapConstruct

setItemIdentifiers(Set) set the itemIdentifiers of the construct

Fields

- protected java.lang.String **firstItemIdentifier**
- protected java.util.Set **itemIdentifiers**

Constructors

- **TopicMapConstruct**
  `public` **TopicMapConstruct**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, is invoked by subtypes to generate a respective object with the given String as itemIdentifier. If the String is not an locator, an URI incorporating an UUID is used as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **addItemIdentifier**
  `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    add an itemIdentifier to a construct. itemIdentifiers have to locators (that is, URIs) and be unique for the TopicMap the construct is part of
  - **Parameters**
    * `itemIdentifier` -- the string to be added as an itemIdentifier
  - **Returns** -- true, if itemIdentifier was successfully added, false if adding failed (because the parameter-string was no locator, i.e. an URI)
- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description**
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant
- **getFirstItemIdentifier**
  `public java.lang.String` **getFirstItemIdentifier**`( )`
  - **Description**
    get the first itemIdentifier of a TopicMapConstruct
  - **Returns** -- firstItemIdentifier
- **getItemIdentifiers**
  `public java.util.Set` **getItemIdentifiers**`( )`
  - **Description**
    get the itemIdentifiers of the construct

  - **Returns** -- the itemIdentifiers of the construct

- **setFirstItemIdentifier**
  `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`

  - **Description**

    set the first itemIdentifier of a TopicMapConstruct

  - **Parameters**

    * `firstItemIdentifier` --

- **setItemIdentifiers**
  `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

  - **Description**

    set the itemIdentifiers of the construct

  - **Parameters**

    * `itemIdentifiers` --

C.1.12 Class TopicName

TopicMap Engine - TopicName TopicNames are constructs which determine the natural language designator of a Topic (and are actually equal to the name(s) of thr represented subject in most of the cases). TopicNames always have an TopicNameType (which is represented using a Topic) - every TopicName of the same nature references the same TopicNameType. As it is not always necessary to define a type for a TopicName, the standard defines a defaultTopicNameType, which has to be used in these cases.

Declaration

public class TopicName
extends ce.tm4scholion.tm.Statement (in C.1.8, page 176)

Field summary

value

Constructor summary

TopicName() default construtor, generates TopicName object with UUID as itemIdentifier
TopicName(String) basic constructor, generates TopicName object with the given String as itemIdentifier

Method summary

addVariant(Variant) add a Variant to the TopicName
checkTMDMCompliance()
equals(Object)
getParent() set the parent element of the TopicName
getType() get the TopicNameType of the TopicName
getValue() get the value of the TopicName (contains the name of the Topic the TopicName is attached to)
getVariants() get the set of Variants of this TopicName
newVariant(String, String, Scope) generate a new Variant for the TopicName with the given value and datatype in the given Scope.
removeVariant(Variant) remove a Variant from the TopicName
setParent(Topic) get the parent element of the TopicName
setType(Topic) set the TopicNameType of the TopicName
setValue(String) set the value of the TopicName (contains the name of the Topic the TopicName is attached to)
setVariants(Set) set the Variants of this TopicName

Fields

- protected java.lang.String **value**

Constructors

- **TopicName**
  `public` **TopicName**`( )`
  - **Description**
    default construtor, generates TopicName object with UUID as itemIdentifier
- **TopicName**
  `public` **TopicName**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates TopicName object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **addVariant**
  `public void` **addVariant**`( Variant` **variant** `)`
  - **Description**
    add a Variant to the TopicName
  - **Parameters**
    * `variant` -- the Variant object to be added
- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM

  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant

- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`

- **getParent**
  `public Topic` **getParent**`( )`

  - **Description**

    set the parent element of the TopicName

  - **Parameters**

    * `parent` -- the Topic the TopicName is bound to

- **getType**
  `public Topic` **getType**`( )`

  - **Description**

    get the TopicNameType of the TopicName

  - **Returns** -- the Topic representing the TopicNameType of the TopicName

- **getValue**
  `public java.lang.String` **getValue**`( )`

  - **Description**

    get the value of the TopicName (contains the name of the Topic the TopicName is attached to)

  - **Returns** -- the value of the TopicName

- **getVariants**
  `public java.util.Set` **getVariants**`( )`

  - **Description**

    get the set of Variants of this TopicName

  - **Returns** -- the set of Variants of this TopicName

- **newVariant**
  `public Variant` **newVariant**`( java.lang.String` **value**`, java.lang.Strin`
  dataType`, Scope` **s** `)`
  - **Description**
    generate a new Variant for the TopicName with the given value and datatype in the given Scope.
  - **Parameters**
    * `value` -- the value of the variant
    * `dataType` -- the datatype of the variant's value
    * `s` -- the Scope the Variant is valid in
  - **Returns** -- the new Variant object
- **removeVariant**
  `public void` **removeVariant**`( Variant` **v** `)`
  - **Description**
    remove a Variant from the TopicName
  - **Parameters**
    * `v` -- the Variant to be removed
- **setParent**
  `protected void` **setParent**`( Topic` **parent** `)`
  - **Description**
    get the parent element of the TopicName
  - **Returns** -- the Topic the TopicName is bound to
- **setType**
  `protected void` **setType**`( Topic` **type** `)`
  - **Description**
    set the TopicNameType of the TopicName
  - **Parameters**
    * `type` -- the Topic representing the TopicNameType of the Topic-Name

- **setValue**
  `public void` **setValue**`( java.lang.String` **value** `)`
  - **Description**
    set the value of the TopicName (contains the name of the Topic the TopicName is attached to)
  - **Returns** -- the value of the TopicName
- **setVariants**
  `public void` **setVariants**`( java.util.Set` **variants** `)`
  - **Description**
    set the Variants of this TopicName
  - **Parameters**
    `*` `variants` --

Members inherited from class `ce.tm4scholion.tm.Statement` (in C.1.8, page 176)

- `public Scope` **getScope**`( )`
- `protected` **scope**
- `public void` **setScope**`( Scope` **s** `)`

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`

- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.1.13 Class Utils

TopicMap Engine - Utils A collection of service routines for internal engine use (not accessible by applications using the engine). Mainly used for type checking and generation of unique IDs. The only items which are accessible externaly are the fields defining the qualifiers for the default dataTypes given in the TMDM (String, IRI, XML)

Declaration

public class Utils
extends java.lang.Object

Field summary

dtIRI the qualifier of the default dataType 'IRI'
dtString the qualifier of the default dataType 'String'
dtXML the qualifier of the default dataType 'XML'

Constructor summary

Utils()

Method summary

getUniqueItemIdentifier() generates an URI representing an unique ID using Java's internal UUID-generation routines (available since Java 5.0)
isDefaultDataType(String) checks if a String holds the URI of a default datatype as defined in the TMDM standard (i.e. a String, a IRI or XML).

isIRIDataType(String) checks if a String holds the URI of the 'IRI' datatype as defined in the TMDM standard.
isLocator(String) checks if a String is a Locator as defined in the TMDM standard (i.e. if the String is an URI).
isStringDataType(String) checks if a String holds the URI of the 'String' datatype as defined in the TMDM standard.
isSuperset(Set, Set) checks if a set of objects is a superset of another set of objects.
isXMLDataType(String) checks if a String holds the URI of the 'XML' datatype as defined in the TMDM standard.
setsContainAtLeastOneEqualElement(Set, Set) checks if two sets of objects contain at least one common equal element.

Fields

- public static final java.lang.String **dtString**
  - the qualifier of the default dataType 'String'
- public static final java.lang.String **dtIRI**
  - the qualifier of the default dataType 'IRI'
- public static final java.lang.String **dtXML**
  - the qualifier of the default dataType 'XML'

Constructors

- **Utils**
  `public` **Utils**`( )`

Methods

- **getUniqueItemIdentifier**
  `protected static java.lang.String` **getUniqueItemIdentifier**`( )`
  - **Description**

generates an URI representing an unique ID using Java's internal UUID-generation routines (available since Java 5.0)

- **Returns** -- an URI of type UUID representing an unique ID

- **isDefaultDataType**
  `protected static boolean` **isDefaultDataType**`( java.lang.String` **s** `)`

  - **Description**
    checks if a String holds the URI of a default datatype as defined in the TMDM standard (i.e. a String, a IRI or XML).
  - **Parameters**
    * `s` -- the String to be checked
  - **Returns** -- true if the String holds the URI of a default datatype, false otherwise

- **isIRIDataType**
  `protected static boolean` **isIRIDataType**`( java.lang.String` **s** `)`

  - **Description**
    checks if a String holds the URI of the 'IRI' datatype as defined in the TMDM standard.
  - **Parameters**
    * `s` -- the String to be checked
  - **Returns** -- true if the String holds the URI of the 'IRI'-datatype, false otherwise

- **isLocator**
  `protected static boolean` **isLocator**`( java.lang.String` **s** `)`

  - **Description**
    checks if a String is a Locator as defined in the TMDM standard (i.e. if the String is an URI). Actually, a Locator format is not explicitly specified in TMDM but is considered to be a URI in this implementation

based on the IRI definition in 'datatypes'. Makes use of a routine of the JENA Semantic Web Framework.

- **Parameters**
  - `s` -- the String to be checked
- **Returns** -- true, if the String is an URI, false otherwise

- **isStringDataType**
  `protected static boolean` **isStringDataType**`( java.lang.String` **s** `)`
  - **Description**
    checks if a String holds the URI of the 'String' datatype as defined in the TMDM standard.
  - **Parameters**
    - `s` -- the String to be checked
  - **Returns** -- true if the String holds the URI of the 'String'-datatype, false otherwise

- **isSuperset**
  `protected static boolean` **isSuperset**`( java.util.Set` **s1**`, java.util.Set` **s2** `)`
  - **Description**
    checks if a set of objects is a superset of another set of objects.
  - **Parameters**
    - `s1` -- the assumed superset
    - `s2` -- the assumed subset
  - **Returns** -- true, if s1 is a superset of s2

- **isXMLDataType**
  `protected static boolean` **isXMLDataType**`( java.lang.String` **s** `)`
  - **Description**
    checks if a String holds the URI of the 'XML' datatype as defined in the TMDM standard.

  - **Parameters**
    - `s` -- the String to be checked
  - **Returns** -- true if the String holds the URI of the 'XML'-datatype, false otherwise

- **setsContainAtLeastOneEqualElement**
  `protected static boolean` **setsContainAtLeastOneEqualElement**`( java.util.Set` **s1**`, java.util.Set` **s2** `)`
  - **Description**
    checks if two sets of objects contain at least one common equal element.
  - **Parameters**
    - `s1` -- the first set to be compared
    - `s2` -- the second set to be compared
  - **Returns** -- true, if the two sets contain at least one common equal element, false otherwise

C.1.14 Class Variant

TopicMap Engine - Variant Variants are alternative forms of a certain TopicName. They can be used to define synonyms for a given topic name but also enable to switch the actual form of representation, as they do not necessarily have to be strings (but can e.g. link to encoded audio). A special use case of variants is that of a sort name, which enables sorting of topics alphabetically (corresponding to unicode order). A Variant is always attached to exactly one TopicName and is the only construct which has to have a mandatory scope definition for standard compliance.

Declaration

public class Variant
extends ce.tm4scholion.tm.Statement (in C.1.8, page 176)

Field summary

dataType
value

Constructor summary

Variant() default construtor, generates Variant object with UUID as itemIdentifier
Variant(String) basic constructor, generates Variant object with the given String as itemIdentifier

Method summary

checkTMDMCompliance()
equals(Object)
getDataType() get the dataType of the Variant
getParent() get the parent element of the Variant
getValue() get the value of the Variant (contains the information ressource representing the Variant or the link to it, respectively)
setDataType(String) set the dataType of the Variant
setParent(TopicName) set the parent element of the Variant
setValue(String) set the value of the Variant (contains the information ressource representing the Variant or the link to it, respectively)

Fields

- protected java.lang.String **value**
- protected java.lang.String **dataType**

Constructors

- **Variant**
  `public` **Variant**`( )`
  - **Description**

default construtor, generates Variant object with UUID as itemIdentifier

- **Variant**
  `public` **Variant**`( java.lang.String` **itemIdentifier** `)`
  - **Description**
    basic constructor, generates Variant object with the given String as itemIdentifier
  - **Parameters**
    * `itemIdentifier` -- has to be unique for the whole TopicMap

Methods

- **checkTMDMCompliance**
  `public abstract java.util.Set` **checkTMDMCompliance**`( )`
  - **Description copied from TopicMapConstruct** (**in C.1.11, page 193**)
    check the compliance of this construct (including all directly attached constructs) to the TMDM
  - **Returns** -- a Set of all checked constructs which do not adhere the TMDM, null is everything is compliant
- **equals**
  `public boolean` **equals**`( java.lang.Object` **argo** `)`
- **getDataType**
  `public java.lang.String` **getDataType**`( )`
  - **Description**
    get the dataType of the Variant
  - **Returns** -- the dataType of the Variant
- **getParent**
  `public TopicName` **getParent**`( )`

  - **Description**
    get the parent element of the Variant
  - **Returns** -- the TopicName the Variant is bound to

- **getValue**
  `public java.lang.String` **getValue**`( )`
  - **Description**
    get the value of the Variant (contains the information ressource representing the Variant or the link to it, respectively)
  - **Returns** -- the value of the Variant

- **setDataType**
  `protected void` **setDataType**`( java.lang.String` **dataType** `)`
  - **Description**
    set the dataType of the Variant
  - **Parameters**
    * `dataType` -- the dataType of the Variant

- **setParent**
  `protected void` **setParent**`( TopicName` **parent** `)`
  - **Description**
    set the parent element of the Variant
  - **Parameters**
    * `parent` -- the TopicName the Variant is bound to

- **setValue**
  `protected void` **setValue**`( java.lang.String` **value** `)`
  - **Description**
    set the value of the Variant (contains the information ressource representing the Variant or the link to it, respectively)
  - **Parameters**
    * `value` -- the value of the Variant

Members inherited from class `ce.tm4scholion.tm.Statement` (in C.1.8, page 176)

- `public Scope` **getScope**`( )`
- `protected` **scope**
- `public void` **setScope**`( Scope` **s** `)`

Members inherited from class `ce.tm4scholion.tm.Reifiable` (in C.1.6, page 170)

- `public Topic` **getReifier**`( )`
- `public void` **setReifier**`( Topic` **reifier** `)`

Members inherited from class `ce.tm4scholion.tm.TopicMapConstruct` (in C.1.11, page 193)

- `public boolean` **addItemIdentifier**`( java.lang.String` **itemIdentifier** `)`
- `public abstract Set` **checkTMDMCompliance**`( )`
- `protected` **firstItemIdentifier**
- `public String` **getFirstItemIdentifier**`( )`
- `public Set` **getItemIdentifiers**`( )`
- `protected` **itemIdentifiers**
- `public void` **setFirstItemIdentifier**`( java.lang.String` **firstItemIdentifier** `)`
- `public void` **setItemIdentifiers**`( java.util.Set` **itemIdentifiers** `)`

C.2 Package ce.tm4scholion.tm.persistency

Package Contents *Page*

Interfaces

TMPersistency 211
Topic Map Engine - Persistency This interface is used to provide generic access to different implementations for making a Topic Map persistent.

C.2.1 Interface TMPersistency

Topic Map Engine - Persistency This interface is used to provide generic access to different implementations for making a Topic Map persistent.

Declaration

public interface TMPersistency

Method summary

connect(String) connect to the data source/sink, from which the TopicMap has to retrieved or has to be stored in
retrieve() retrieve a TopicMap from the connected data source
store(TopicMap) store the given TopicMap into the connected data sink

Methods

- **connect**
  `boolean` **connect**`( java.lang.String` **connection** `)`
  - **Description**
    connect to the data source/sink, from which the TopicMap has to be retrieved or has to be stored in
  - **Parameters**
    * `connection` -- a string adressing the data source/sink to connect to. Concrete format is specified by implementations implementing this interface.
  - **Returns** -- true, if successfully connected, false otherwise
- **retrieve**
  `ce.tm4scholion.tm.TopicMap` **retrieve**`( )`
  - **Description**
    retrieve a TopicMap from the connected data source

  - **Returns** -- the retrieved TopicMap

- **store**
  `boolean` **store**`( ce.tm4scholion.tm.TopicMap` **tm** `)`
  - **Description**
    store the given TopicMap into the connected data sink
  - **Parameters**
    * `tm` -- the TopicMap to be stored
  - **Returns** -- true, if storing was successful, false otherwise

Appendix D

OL Content Models

In this annex, the structure and usage of the Content Models is described briefly. The figures used here have been generated using the Doxygen documentation suite (which itself makes use of the GraphViz-toolset). Figure D.1 gives an overview about the structure of the currently implemented content models for learning and communication content as well as the common elements. In the following sections, a more detailed overview about the implemented content elements will be given (generated directly from source).

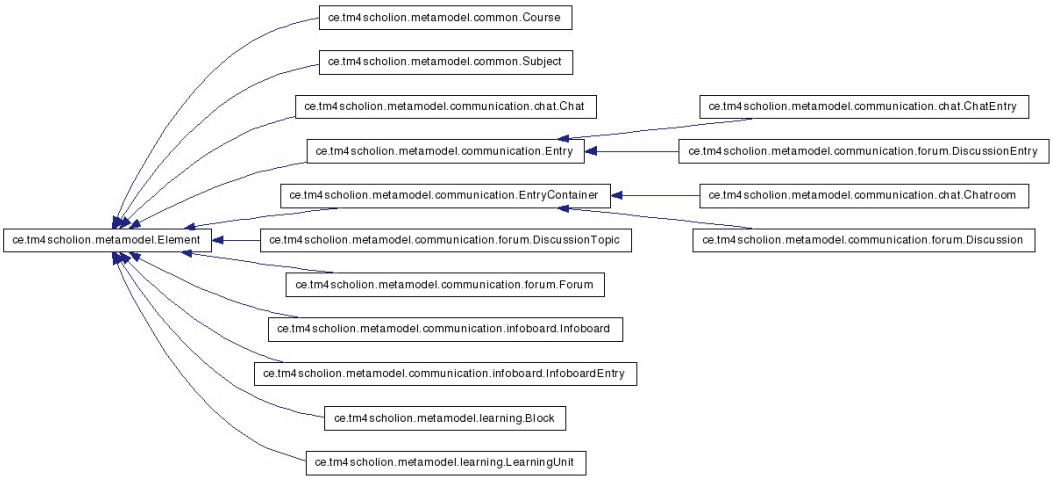

Figure D.1: Overview of currently implemented content model structure

D.1 Package ce.tm4scholion.metamodel

Package Contents *Page*

Classes

Element 214
The Element class is the basic class for all content elements used in the OL learning models.
Manager 218
Metamodel Manager - this class provides access methods to generate and manage meta-models of content management.
Manager.RoleElementCombination 225
A wrapper class to represent combinations of elements and roles to be used in associations

D.1.1 Class Element

The Element class is the basic class for all content elements used in the OL learning models. An element in minimum has a name and always may have an author and/or an owner assigned. Every element is represented by exactly one topic in the topic map (adressed using the myRep-field). It is however not sufficient to simply output this topic to map the element onto the topic map. The contained associations and attached or embedded elements also have to be considered. The method toTopicMap covers this functionality and has to be extended for every actual element type implemented in the meta models.

Declaration

public abstract class Element
extends java.lang.Object

All known subclasses

Subject (in D.2.3, page 230), Course (in D.2.1, page 227), LearningUnit (in D.3.2, page 235), Block (in D.3.1, page 232), EntryContainer (in D.4.2, page 239), Entry (in D.4.1, page 238), Chatroom (in D.5.3, page 247), ChatEntry (in D.5.2, page 246), Chat (in

D.5.1, page 244), Forum (in D.6.4, page 253), DiscussionTopic (in D.6.3, page 251), DiscussionEntry (in D.6.2, page 250), Discussion (in D.6.1, page 249), InfoboardEntry (in D.7.2, page 255), Infoboard (in D.7.1, page 254)

Field summary

author
mgr
myRep
name
owner

Constructor summary

Element(String, Manager) Constructor to create a new element from scratch
Element(Topic, Manager) Constructor to reconstuct the element from an already existing topic map

Method summary

getAuthor() returns the author of this element as a Subject object
getName() returns the name of the element
getOwner() returns the owner of this element as a Subject object
getRep() return the topic used to represent this element.
setAuthor(Subject) set the author of this element
setOwner(Subject) set the owner of this element
toTopicMap() map a default element to the underlying topic map.

Fields

- protected Manager **mgr**
- protected java.lang.String **name**
- protected common.Subject **author**
- protected common.Subject **owner**

- protected ce.tm4scholion.tm.Topic **myRep**

Constructors

- **Element**
  `public` **Element**`( java.lang.String` **name**`, Manager` **mgr** `)`
  - **Description**
    Constructor to create a new element from scratch
  - **Parameters**
    * `name` -- the name of the new element
    * `mgr` -- the manager of the model this element is contained in
- **Element**
  `public` **Element**`( ce.tm4scholion.tm.Topic` **myRep**`, Manager` **mgr** `)`
  - **Description**
    Constructor to reconstuct the element from an already existing topic map
  - **Parameters**
    * `myRep` -- the topic representing the element to be created
    * `mgr` -- the manager of the model this element is contained in

Methods

- **getAuthor**
  `public common.Subject` **getAuthor**`( )`
  - **Description**
    returns the author of this element as a Subject object
  - **Returns** -- the Subject element representing the author of this element
- **getName**
  `public java.lang.String` **getName**`( )`

  - **Description**

    returns the name of the element

  - **Returns** -- the name of the element

- **getOwner**
  `public common.Subject` **getOwner**`( )`

  - **Description**

    returns the owner of this element as a Subject object

  - **Returns** -- the Subject element representing the owner of this element

- **getRep**
  `public ce.tm4scholion.tm.Topic` **getRep**`( )`

  - **Description**

    return the topic used to represent this element. If the topic not yet exists, one is created.

  - **Returns** -- the topic representing this element

- **setAuthor**
  `public void` **setAuthor**`( common.Subject` **author** `)`

  - **Description**

    set the author of this element

  - **Parameters**

    * `author` -- the Subject element representing the author

- **setOwner**
  `public void` **setOwner**`( common.Subject` **owner** `)`

  - **Description**

    set the owner of this element

  - **Parameters**

    * `owner` -- the Subject element representing the owner

- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description**
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

D.1.2 Class Manager

Metamodel Manager - this class provides access methods to generate and manage meta-models of content management.

Declaration

public class Manager
extends java.lang.Object

Constructor summary

Manager() Constructor of the meta-model manager class.

Method summary

addAssociation(String, Set, Set) add an association between defined topics in specified roles and - if necessary - a certain scope
addMetamodelAssociation(String, Set) register a new association between defined modeling elements.
addMetamodelElement(String) register a new element for a certain meta model.
addResource(Topic, String, String, Set) add resource data of a certain type to a specified topic - if necessary, set a scope
addResourceTypesForElement(Topic, Set) register new resource types (occurrences) for a defined modeling elements.
concretizeRole(String, Set) allows to specify subroles (or concrete roles) which can be inserted instead of the given top-level role in any association type it is used in.

createNewNestedAssocType(Set, Set) shortcut to define a new assocation type named 'nested', which is needed regularly to define hierarchical relationships

createNewNestedAssocType(Set, Set, Topic) shortcut to define a new assocation type named 'nested', which is needed regularly to define hierarchical relationships.

createNewSiblingAssocType(Set) shortcut to define a new assocation type named 'sibling', which is needed regularly to define ordered sets of elements

getCommonManager() retrieve the manager for common content elements

getCommunicationManager() retrieve the manager for communication content

getLearningManager() retrieve the manager for learning content

getMetamodelElement(String) retrieve the topic representing a certain meta element by name

getTMManager() retrieve the manager of the underlying topic map

instantiateElement(String, Topic) creates a topic representing an actual element of a certain type

validateAssociation(String, Set) validates, wether a given set of topic-role-combinations can be used in a certain assocation.

validateResource(Topic, String) validates, wether a given element (represented by a topic) of a certain type can contain a resource of a certain type.

Constructors

- **Manager**
  `public` **Manager**`( )`

  – **Description**

  Constructor of the meta-model manager class. Also creates and administrates the manager classes for the contained meta-models.

Methods

- **addAssociation**
  `public boolean` **addAssociation**`( java.lang.String` **assocType**`, java.util.Set` **rt**`, java.util.Set` **scope** `)`
  - **Description**
    add an association between defined topics in specified roles and - if necessary - a certain scope
  - **Parameters**
    * `assocType` -- the name of the association type to be used
    * `rt` -- the set of topics to be associated in the assigned roles
    * `scope` -- a set of topics defining the scope of this association
  - **Returns** --
- **addMetamodelAssociation**
  `public void` **addMetamodelAssociation**`( java.lang.String` **name**`, java.util.Set` **validRoleElementCombinations** `)`
  - **Description**
    register a new association between defined modeling elements. Every type of association that is used to link content of any kind has to be registered using this method. In addition, the types of elements to be linked and their respective roles have to be specified.
  - **Parameters**
    * `name` -- the name of the new association type
    * `validRoleElementCombinations` -- a set of Role-Element-Combinations which specify the element types that can be linked with this association in certain roles
- **addMetamodelElement**
  `public ce.tm4scholion.tm.Topic` **addMetamodelElement**`( java.lang.St`
  name `)`
  - **Description**

register a new element for a certain meta model. Every type of element that is used to represent content of any kind has to be registered using this method

- **Parameters**
  - `name` -- the name of the new meta model element
- **Returns** -- the topic representing the new meta element

- **addResource**
  `public boolean` **addResource**`( ce.tm4scholion.tm.Topic` **element**`, java.lang.String` **data**`, java.lang.String` **type**`, java.util.Set` **scope** `)`
  - **Description**
    add resource data of a certain type to a specifed topic - if necessary, set a scope
  - **Parameters**
    - `element` -- the topic representing the element to which the resource has to be added
    - `data` -- the actual resource data (either a link, or raw or xml content
    - `type` -- the type of the resource
    - `scope` -- a set of topics defining the scope of this association
  - **Returns** --

- **addResourceTypesForElement**
  `public void` **addResourceTypesForElement**`( ce.tm4scholion.tm.Topic` **element**`, java.util.Set` **resourceTypes** `)`
  - **Description**
    register new resource types (occurrences) for a defined modeling elements. Every type of resource that is used to instantiate the resprective element has to be registered using this method.
  - **Parameters**
    - `element` -- the topic representing the meta-element, to which the resource types is added

    * `resourceTypes` -- a set of names for the new ressource types to be added

- **concretizeRole**
  `public void` **concretizeRole**`( java.lang.String` **role**`, java.util.Set` **concreteRoles** `)`
  - **Description**
    allows to specify subroles (or concrete roles) which can be inserted instead of the given top-level role in any association type it is used in. This allows to specify roles that refine a given role while assuring that associations using this role are still verified correctly when using one of the concrete, refined role types.
  - **Parameters**
    * `role` -- the role to be refined with concrete roles
    * `concreteRoles` -- a set of names for the concrete roles
- **createNewNestedAssocType**
  `public void` **createNewNestedAssocType**`( java.util.Set` **superordinates**`, java.util.Set` **subordinates** `)`
  - **Description**
    shortcut to define a new assocation type named 'nested', which is needed regularly to define hierarchical relationships
  - **Parameters**
    * `superordinates` -- the element types to be used on the upper layer of the hierarchy
    * `subordinates` -- the element types to be used on the lower layer of the hierarchy
- **createNewNestedAssocType**
  `public void` **createNewNestedAssocType**`( java.util.Set` **superordinates**`, java.util.Set` **subordinates**`, ce.tm4scholion.tm.Topic` **firstSubelement** `)`
  - **Description**

shortcut to define a new assocation type named 'nested', which is needed regularly to define hierarchical relationships. In this case the, subordinates are an ordered set and thus have a fist element

- **Parameters**
  - `superordinates` -- the element types to be used on the upper layer of the hierarchy
  - `subordinates` -- the element types to be used on the lower layer of the hierarchy
  - `firstSubelement` -- the element type for the first element of the lower layer of the hierarchy

- **createNewSiblingAssocType**
  `public void` **createNewSiblingAssocType**`( java.util.Set` **elements** `)`
  - **Description**
    shortcut to define a new assocation type named 'sibling', which is needed regularly to define ordered sets of elements
  - **Parameters**
    - `elements` -- the types of elements that can be used in the ordered set of elements

- **getCommonManager**
  `public common.Manager` **getCommonManager**`( )`
  - **Description**
    retrieve the manager for common content elements
  - **Returns** -- the manager for common content elements

- **getCommunicationManager**
  `public communication.Manager` **getCommunicationManager**`( )`
  - **Description**
    retrieve the manager for communication content
  - **Returns** -- the manager for communication content

- **getLearningManager**
  `public learning.Manager` **getLearningManager**`( )`
  - **Description**
    retrieve the manager for learning content
  - **Returns** -- the manager for learning content
- **getMetamodelElement**
  `public ce.tm4scholion.tm.Topic` **getMetamodelElement**`( java.lang.Str` **name** `)`
  - **Description**
    retrieve the topic representing a certain meta element by name
  - **Parameters**
    * `name` -- the name of the meta element to retrieve
  - **Returns** -- the topic representing the meta element, false if not found
- **getTMManager**
  `public ce.tm4scholion.tm.Manager` **getTMManager**`( )`
  - **Description**
    retrieve the manager of the underlying topic map
  - **Returns** -- the manager object of the underlying topic map
- **instantiateElement**
  `public ce.tm4scholion.tm.Topic` **instantiateElement**`( java.lang.String` **name**`, ce.tm4scholion.tm.Topic` **elementType** `)`
  - **Description**
    creates a topic representing an actual element of a certain type
  - **Parameters**
    * `name` -- the name of the instantiated element
    * `elementType` -- the topic representing the element type of the new element
  - **Returns** --

- **validateAssociation**
  `public boolean` **validateAssociation**`( java.lang.String` **assocType**`, java.util.Set` **rt** `)`
  - **Description**
    validates, wether a given set of topic-role-combinations can be used in a certain assocation. The validation is based on the specified association types
  - **Parameters**
    * `assocType` -- the association type for which the validation is carried out
    * `rt` -- the topic-role combination of actual elements to be validated
  - **Returns** --

- **validateResource**
  `public boolean` **validateResource**`( ce.tm4scholion.tm.Topic` **element**`, java.lang.String` **type** `)`
  - **Description**
    validates, wether a given element (represented by a topic) of a certain type can contain a resource of a certain type. The validation is based on the specified resource types
  - **Parameters**
    * `element` -- the element to be validated
    * `type` -- the resource type for which the validation is carried out
  - **Returns** --

D.1.3 Class Manager.RoleElementCombination

A wrapper class to represent combinations of elements and roles to be used in associations

Declaration

public static class Manager.RoleElementCombination
extends java.lang.Object

Field summary

cardinality
elements
role

Constructor summary

Manager.RoleElementCombination(String, Set, String)

Fields

- public java.lang.String **role**
- public java.util.Set **elements**
- public java.lang.String **cardinality**

Constructors

- **Manager.RoleElementCombination**
  `public` **Manager.RoleElementCombination** `( java.lang.String` **role**`, java.util.Set` **elements**`, java.lang.String` **cardinality** `)`

D.2 Package ce.tm4scholion.metamodel.common

Package Contents *Page*

Classes

Course 227

Manager 229

Subject 230

D.2.1 Class Course

Declaration

public class Course
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Course(Topic, Manager)

Method summary

addChat(Chat)
addForum(Forum)
getContainedChats()
getContainedForums()
getContainedInfoboard()
getLUs()
setContainedChats(Set)
setContainedForums(Set)
setContainedInfoboard(Infoboard)
setLUs(Vector)
toTopicMap()

Constructors

- **Course**
  `public` **Course**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamode`
  mgr `)`

Methods

- **addChat**
  `public void` **addChat**`( ce.tm4scholion.metamodel.communication.chat.Chat`
  chat `)`

- **addForum**
  `public void` **addForum**`( ce.tm4scholion.metamodel.communication.for`
  forum `)`
- **getContainedChats**
  `public java.util.Set` **getContainedChats**`( )`
- **getContainedForums**
  `public java.util.Set` **getContainedForums**`( )`
- **getContainedInfoboard**
  `public ce.tm4scholion.metamodel.communication.infoboard.Infoboard`
  getContainedInfoboard`( )`
- **getLUs**
  `public java.util.Vector` **getLUs**`( )`
- **setContainedChats**
  `public void` **setContainedChats**`( java.util.Set` **contained-Chats** `)`
- **setContainedForums**
  `public void` **setContainedForums**`( java.util.Set` **contained-Forums** `)`
- **setContainedInfoboard**
  `public void` **setContainedInfoboard**`( ce.tm4scholion.metamodel.commun`
  containedInfoboard `)`
- **setLUs**
  `public void` **setLUs**`( java.util.Vector` **lus** `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.2.2 Class Manager

Declaration

public class Manager
extends java.lang.Object

Field summary

course
subject

Constructor summary

Manager(Manager)

Method summary

discuss(Element, Element)
generateCourse(String)
generateSubject(String)

Fields

- public ce.tm4scholion.tm.Topic **subject**
- public ce.tm4scholion.tm.Topic **course**

Constructors

- **Manager**
  `public` **Manager**`( ce.tm4scholion.metamodel.Manager` **tmManager** `)`

Methods

- **discuss**
  `public boolean` **discuss**`( ce.tm4scholion.metamodel.Element` **discussionObject**`, ce.tm4scholion.metamodel.Element` **discussedIn** `)`
- **generateCourse**
  `public Course` **generateCourse**`( java.lang.String` **name** `)`
- **generateSubject**
  `public Subject` **generateSubject**`( java.lang.String` **name** `)`

D.2.3 Class Subject

Declaration

public class Subject
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Subject(Topic, Manager)

Method summary

toTopicMap()

Constructors

- **Subject**
  `public` **Subject**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamode`
  mgr `)`

Methods

- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.3 Package ce.tm4scholion.metamodel.learning

Package Contents *Page*

Classes

Block 232

LearningUnit 235

Manager 236

D.3.1 Class Block

Declaration

public class Block
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Block(String, Manager)
Block(Topic, Manager)

Method summary

addAnnotation(LearningUnit, String, Subject, String, String)
addContent(LearningUnit, String, String)
getAnnotation(LearningUnit, String, Subject, String)
getBlocks(LearningUnit)
getContent(LearningUnit, String)
getContentTypeInLearningUnit(LearningUnit)
setContainedBlocks(Vector, LearningUnit)
setContentTypeInLearningUnit(String, LearningUnit)
toTopicMap()

toTopicMap(LearningUnit)

Constructors

- **Block**
  `public` **Block**`( java.lang.String` **name**`, ce.tm4scholion.metamodel.Manager` **mgr** `)`
- **Block**
  `public` **Block**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamodel` **mgr** `)`

Methods

- **addAnnotation**
  `public void` **addAnnotation**`( LearningUnit` **lu**`, java.lang.String` **lod**`, ce.tm4scholion.metamodel.common.Subject` **author**`, java.lang.String` **type**`, java.lang.String` **data** `)`
- **addContent**
  `public void` **addContent**`( LearningUnit` **lu**`, java.lang.String` **lod**`, java.lang.String` **data** `)`
- **getAnnotation**
  `public java.lang.String` **getAnnotation**`( LearningUnit` **lu**`, java.lang.String` **lod**`, ce.tm4scholion.metamodel.common.Subject` **author**`, java.lang.String` **type** `)`
- **getBlocks**
  `public java.util.Vector` **getBlocks**`( LearningUnit` **lu** `)`
- **getContent**
  `public java.lang.String` **getContent**`( LearningUnit` **lu**`, java.lang.String` **lod** `)`
- **getContentTypeInLearningUnit**
  `public java.lang.String` **getContentTypeInLearningUnit**`( LearningUnit` **lu** `)`

- **setContainedBlocks**
  `public void` **setContainedBlocks**`( java.util.Vector` **blocks**`, LearningUnit` **inScope** `)`

- **setContentTypeInLearningUnit**
  `public void` **setContentTypeInLearningUnit**`( java.lang.String` **contentType**`, LearningUnit` **lu** `)`

- **toTopicMap**
  `public void` **toTopicMap**`( )`

  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)

    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

- **toTopicMap**
  `public void` **toTopicMap**`( LearningUnit` **lu** `)`

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.3.2 Class LearningUnit

Declaration

public class LearningUnit
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

LearningUnit(String, Manager)
LearningUnit(Topic, Manager)

Method summary

getBlocks()
setContainedBlocks(Vector)
toTopicMap()

Constructors

- **LearningUnit**
  `public` **LearningUnit**`( java.lang.String` **name**`, ce.tm4scholion.metamodel.M`
  mgr `)`
- **LearningUnit**
  `public` **LearningUnit**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.met`
  mgr `)`

Methods

- **getBlocks**
  `public java.util.Vector` **getBlocks**`( )`
- **setContainedBlocks**
  `public void` **setContainedBlocks**`( java.util.Vector` **blocks**
  `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`

– **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.3.3 Class Manager

Declaration

public class Manager
extends java.lang.Object

Field summary

block
learningUnit

Constructor summary

Manager(Manager)

Method summary

generateBlock(String)
generateLearningUnit(String)

Fields

- public ce.tm4scholion.tm.Topic **block**
- public ce.tm4scholion.tm.Topic **learningUnit**

Constructors

- **Manager**
  `public` **Manager**`( ce.tm4scholion.metamodel.Manager` **tmManager** `)`

Methods

- **generateBlock**
  `public Block` **generateBlock**`( java.lang.String` **name** `)`
- **generateLearningUnit**
  `public LearningUnit` **generateLearningUnit**`( java.lang.String` **name** `)`

D.4 Package ce.tm4scholion.metamodel.communication

Package Contents *Page*

Classes

Entry 238

EntryContainer 239

Manager 241

D.4.1 Class Entry

Declaration

public abstract class Entry
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

All known subclasses

ChatEntry (in D.5.2, page 246), DiscussionEntry (in D.6.2, page 250)

Field summary

data

Constructor summary

Entry(String, Manager)
Entry(Topic, Manager)

Method summary

setText(String)
toTopicMap()

Fields

- protected java.lang.String **data**

Constructors

- **Entry**
  `public` **Entry**`( java.lang.String` **name**`, ce.tm4scholion.metamodel.Mar`
  mgr `)`
- **Entry**
  `public` **Entry**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metar`
  mgr `)`

Methods

- **setText**
  `public void` **setText**`( java.lang.String` **data** `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.4.2 Class EntryContainer

Declaration

public abstract class EntryContainer
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

All known subclasses

Chatroom (in D.5.3, page 247), Discussion (in D.6.1, page 249)

Field summary

data
entries

Constructor summary

EntryContainer(String, Manager)
EntryContainer(Topic, Manager)

Method summary

addDescription(String)
addEntry(Entry)
getEntries()
setEntries(Vector)
toTopicMap()

Fields

- protected java.util.Vector **entries**
- protected java.lang.String **data**

Constructors

- **EntryContainer**
  `public` **EntryContainer**`( java.lang.String` **name**`, ce.tm4scholion.meta`
  mgr `)`
- **EntryContainer**
  `public` **EntryContainer**`( ce.tm4scholion.tm.Topic` **myRep**`,`
  `ce.tm4scholion.metamodel.Manager` **mgr** `)`

Methods

- **addDescription**
  `public void` **addDescription**`( java.lang.String` **data** `)`

- **addEntry**
  `public void` **addEntry**`( Entry` **entry** `)`
- **getEntries**
  `public java.util.Vector` **getEntries**`( )`
- **setEntries**
  `public void` **setEntries**`( java.util.Vector` **entries** `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.4.3 Class Manager

Declaration

public class Manager
extends java.lang.Object

Field summary

chat
chatEntry
chatRoom
discussion
discussionEntry
discussionTopic
forum
infoboard
infoboardEntry

Constructor summary

Manager(Manager)

Method summary

generateChat(String)
generateChatEntry(String)
generateChatroom(String)
generateDiscussion(String)
generateDiscussionEntry(String)
generateDiscussionTopic(String)
generateForum(String)
generateInfoboard(String)
generateInfoboardEntry(String)

Fields

- public ce.tm4scholion.tm.Topic **forum**
- public ce.tm4scholion.tm.Topic **discussionTopic**
- public ce.tm4scholion.tm.Topic **discussion**
- public ce.tm4scholion.tm.Topic **discussionEntry**
- public ce.tm4scholion.tm.Topic **chat**

- public ce.tm4scholion.tm.Topic **chatRoom**
- public ce.tm4scholion.tm.Topic **chatEntry**
- public ce.tm4scholion.tm.Topic **infoboard**
- public ce.tm4scholion.tm.Topic **infoboardEntry**

Constructors

- **Manager**
  `public` **Manager**`( ce.tm4scholion.metamodel.Manager` **tmManager** `)`

Methods

- **generateChat**
  `public chat.Chat` **generateChat**`( java.lang.String` **name** `)`

- **generateChatEntry**
  `public Entry` **generateChatEntry**`( java.lang.String` **name** `)`

- **generateChatroom**
  `public chat.Chatroom` **generateChatroom**`( java.lang.String` **name** `)`

- **generateDiscussion**
  `public forum.Discussion` **generateDiscussion**`( java.lang.String` **name** `)`

- **generateDiscussionEntry**
  `public forum.DiscussionEntry` **generateDiscussionEntry**`( java.lang.String` **name** `)`

- **generateDiscussionTopic**
  `public forum.DiscussionTopic` **generateDiscussionTopic**`( java.lang.String` **name** `)`

- **generateForum**
  `public forum.Forum` **generateForum**`( java.lang.String` **name** `)`
- **generateInfoboard**
  `public infoboard.Infoboard` **generateInfoboard**`( java.lang.String` **name** `)`
- **generateInfoboardEntry**
  `public infoboard.InfoboardEntry` **generateInfoboardEntry**`( java.lang.String` **name** `)`

D.5 Package ce.tm4scholion.metamodel.communicati

Package Contents *Page*

Classes

Chat 244

ChatEntry 246

Chatroom 247

D.5.1 Class Chat

Declaration

public class Chat
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Chat(String, Manager)
Chat(Topic, Manager)

Method summary

addChatRoom(Chatroom)
getChatRooms()
setChatRooms(Set)
toTopicMap()

Constructors

- **Chat**
  `public` **Chat**`( java.lang.String` **name**`, ce.tm4scholion.metamodel.Manager` **mgr** `)`

- **Chat**
  `public` **Chat**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamodel.M` **mgr** `)`

Methods

- **addChatRoom**
  `public void` **addChatRoom**`( Chatroom` **chatRoom** `)`

- **getChatRooms**
  `public java.util.Set` **getChatRooms**`( )`

- **setChatRooms**
  `public void` **setChatRooms**`( java.util.Set` **chatRooms** `)`

- **toTopicMap**
  `public void` **toTopicMap**`( )`

  – **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)

  map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.5.2 Class ChatEntry

Declaration

public class ChatEntry
extends ce.tm4scholion.metamodel.communication.Entry (in D.4.1, page 238)

Constructor summary

ChatEntry(Topic, Manager)

Constructors

- **ChatEntry**
  `public` **ChatEntry**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.m`
  mgr `)`

Members inherited from class `ce.tm4scholion.metamodel.communication.Entr` (in D.4.1, page 238)

- `protected` **data**
- `public void` **setText**`( java.lang.String` **data** `)`
- `public void` **toTopicMap**`( )`

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.5.3 Class Chatroom

Declaration

public class Chatroom
extends ce.tm4scholion.metamodel.communication.EntryContainer (in D.4.2, page 239)

Constructor summary

Chatroom(Topic, Manager)

Method summary

toTopicMap()

Constructors

- **Chatroom**
  `public` **Chatroom**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamo`
  mgr `)`

Methods

- **toTopicMap**
  `public void` **toTopicMap**`( )`

  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.communication.Entr` (in D.4.2, page 239)

- `public void` **addDescription**`( java.lang.String` **data** `)`
- `public void` **addEntry**`( Entry` **entry** `)`
- `protected` **data**
- `protected` **entries**
- `public Vector` **getEntries**`( )`
- `public void` **setEntries**`( java.util.Vector` **entries** `)`
- `public void` **toTopicMap**`( )`

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.6 Package ce.tm4scholion.metamodel.communication.f

Package Contents *Page*

Classes

Discussion 249

DiscussionEntry 250

DiscussionTopic 251

Forum 253

D.6.1 Class Discussion

Declaration

public class Discussion
extends ce.tm4scholion.metamodel.communication.EntryContainer (in D.4.2, page 239)

Constructor summary

Discussion(Topic, Manager)

Constructors

- **Discussion**
  `public` **Discussion**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metam`
  mgr `)`

Members inherited from class `ce.tm4scholion.metamodel.communication.EntryCont` (in D.4.2, page 239)

- `public void` **addDescription**`( java.lang.String` **data** `)`

- `public void` **addEntry**`( Entry` **entry** `)`
- `protected` **data**
- `protected` **entries**
- `public Vector` **getEntries**`( )`
- `public void` **setEntries**`( java.util.Vector` **entries** `)`
- `public void` **toTopicMap**`( )`

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.6.2 Class DiscussionEntry

Declaration

public class DiscussionEntry
extends ce.tm4scholion.metamodel.communication.Entry (in D.4.1, page 238)

Constructor summary

DiscussionEntry(Topic, Manager)

Constructors

- **DiscussionEntry**
  `public` **DiscussionEntry**`( ce.tm4scholion.tm.Topic` **myRep**`,`
  `ce.tm4scholion.metamodel.Manager` **mgr** `)`

Members inherited from class `ce.tm4scholion.metamodel.communication.Entry`
(in D.4.1, page 238)

- `protected` **data**
- `public void` **setText**`( java.lang.String` **data** `)`
- `public void` **toTopicMap**`( )`

Members inherited from class `ce.tm4scholion.metamodel.Element`
(in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.6.3 Class DiscussionTopic

Declaration

public class DiscussionTopic
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

DiscussionTopic(Topic, Manager)

Method summary

addDiscussion(Discussion)
toTopicMap()

Constructors

- **DiscussionTopic**
  `public` **DiscussionTopic**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamodel.Manager` **mgr** `)`

Methods

- **addDiscussion**
  `public void` **addDiscussion**`( Discussion` **d** `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.6.4 Class Forum

Declaration

public class Forum
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Forum(Topic, Manager)

Method summary

addDiscussionTopic(DiscussionTopic)
toTopicMap()

Constructors

- **Forum**
  `public` **Forum**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamodel`
  mgr `)`

Methods

- **addDiscussionTopic**
  `public void` **addDiscussionTopic**`( DiscussionTopic` **dt** `)`
- **toTopicMap**
  `public void` **toTopicMap**`( )`
  - **Description copied from ce.tm4scholion.metamodel.Element** (**in D.1.1, page 214**)
    map a default element to the underlying topic map. This includes attaching author and/or owner elements, if available.

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**

- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.7 Package ce.tm4scholion.metamodel.communicati

Package Contents *Page*

Classes

Infoboard 254

InfoboardEntry 255

D.7.1 Class Infoboard

Declaration

public class Infoboard
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

Infoboard(Topic, Manager)

Constructors

- **Infoboard**
  `public` **Infoboard**`( ce.tm4scholion.tm.Topic` **myRep**`, ce.tm4scholion.metamo`
  mgr `)`

Members inherited from class `ce.tm4scholion.metamodel.Element`
(in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`

D.7.2 Class InfoboardEntry

Declaration

public class InfoboardEntry
extends ce.tm4scholion.metamodel.Element (in D.1.1, page 214)

Constructor summary

InfoboardEntry(Topic, Manager)

Constructors

- **InfoboardEntry**
  `public` **InfoboardEntry**`( ce.tm4scholion.tm.Topic` **myRep**`,`
  `ce.tm4scholion.metamodel.Manager` **mgr** `)`

Members inherited from class `ce.tm4scholion.metamodel.Element` (in D.1.1, page 214)

- `protected` **author**
- `public Subject` **getAuthor**`( )`
- `public String` **getName**`( )`
- `public Subject` **getOwner**`( )`
- `public Topic` **getRep**`( )`
- `protected` **mgr**
- `protected` **myRep**
- `protected` **name**
- `protected` **owner**
- `public void` **setAuthor**`( common.Subject` **author** `)`
- `public void` **setOwner**`( common.Subject` **owner** `)`
- `public void` **toTopicMap**`( )`