UNIVERSITY OF OSLO
Department of informatics

Merging Topic Maps on Mobile Devices

Master thesis
60 credits

Joril B. Andersen

8. November 2006

Preface

This thesis has been a long journey, from the first sketched out suggestion for a master thesis to a well-defined problem area and an actual implementation. The first presentation of the thesis included a description of an emergency scenario that could be more effective with the use of ontologies for information sharing. The scenario included usage of a Mobile Ad-hoc Network and the objective was to exchange knowledge across organizations. The suggested master thesis was a part of the knowledge management field in the Ad-hoc InfoWare project, and this included the usage of RDF and OWL. I suggested Topic Maps, which is a technology for knowledge representation that can express ontologies. The standard is used as a knowledge layer between the information resources and the user interface. Topic maps can, in addition, be merged, a mechanism that is of great interest in distributed systems. These qualifications seemed very suitable for the requirements posed by the Ad-hoc InfoWare project. The Topic Maps standard was selected and the process of finding suitable Topic Maps engines started. It was also important to acquire a very well understanding of the Topic Maps standard, which proved to be a challenge. To jump-start this I managed to get a free attendance at a Topic Maps course held by Ontopia.

In the first 4 months I worked fairly close with Samuel Vigdal who wrote a short master thesis on a closely related subject. His task was to create a Topic Maps laboratory where I could toy around and test the use of Topic Maps on different resource-weak devices. We used many hours on discussions and we tried to figure out a suitable use case scenario where Topic Maps had a utility value. From these results we found appropriate tests and queries to be performed in the laboratory. We elaborated ontologies and topic maps, and a use case to back up the thesis. The tables turned and Samuel found a mobile Topic Map engine (MTV) which he used to test the queries on different mobile devices. This changed the course of my thesis and I decided to improve the MTV by implementing a functionality that was lacking; the merging of topic maps. The Topic Maps API contains a merging method that should be implemented in a Topic Map engine. The merging method ensures that identical information items are merged to avoid redundancy. Merging expands the knowledge layer and the ultimate goal is “Seamless Knowledge”.

I started mapping the Topic Maps community in search of other implementations of a mobile Topic Maps engine or a similar project. The MTV turned out to be the only existing engine for *mobile devices*. A closer communication with the responsible people behind the MTV project was established and they agreed to let me continue the work. An open source implementation, the tinyTIM was used as a study-case, in addition to the Topic Maps XTM version 1.0. After studying the tinyTIM, the MTV was next. This engine, since it was still quite new, had no documentations in form of models, class diagrams, or descriptions, except java documentations. The challenge was being acquainted with the code in all aspects to be able to extend the engine with merging capabilities. In addition, a study of the Java ME was needed. A draft of the Topic Maps data model 1.0 was released during the summer. The data model contained a more detailed merging description and contributed to a better understanding of the merging process. Since the implementation was intended for mobile phones, I developed a complete merge method and a lightweight edition that requires fewer resources. After the implementation, I performed benchmark tests with different sized topic maps.

I would like to thank my supervisor, Ellen Munthe-Kaas, for support and productive discussions. She has guided me along the way and taken time to meet me regularly every week. She has also been very helpful in given me advice on the elaboration of the thesis. The co-operation with Samuel Vigdal was very useful, and contributed to several new ideas and reflections around the progression of the thesis. In addition, the collaboration resulted in the

elaboration of an ontology and example topic maps that have been used for testing purposes. Members of the Topic Maps mailing list have also been very friendly and helpful when I was at a loss.

I would like to thank my friends who never gave up on me when I turned into an unsocial geek and tried to explain the concept of merging topic maps at every available occasion.

Last but not least, I would like to thank my parents for unconditional support and encourage when I needed it.

Abstract

This thesis is a part of the knowledge module in the Ad-Hoc InfoWare project at the University of Oslo. The goal of the project is to enable information sharing between rescue personnel at an emergency scenario by the use of shared ontologies. During a rescue operation, there are different organizations involved like police, ambulance and fire fighters. To ensure an efficient operation they need to co-operate across organizational boundaries. This creates the need for information sharing between different systems with different interfaces in a mobile ad-hoc network.

Topic Maps can represent ontologies and can be realised as a knowledge layer on top of different system in different domains. The Topic Map layer will serve as a common ground for all the systems and enable the different laptops, PDAs and mobile devices to share information. The Topic Maps technology has many functions that can contribute to the desired result. The merging mechanism is one of the cornerstones in the Topic Map standard. This mechanism removes redundant information items when two arbitrary topic maps merge. This can contribute to a global view of the different domains and tie them together across the different systems.

So far, there has hardly existed any application of Topic Maps on mobile devices such as cellular phones. The Mobile Topic Viewer (MTV) is the only engine for mobile devices. There is one important aspect missing in the MTV, the process of merging topic maps. To test merging of topic maps on mobile devices, the concept of merging has been implemented in the MTV. Since the Topic Maps standard is targeting usage on resourceful devices, like laptops and PC's, and not on mobile devices, I performed benchmark tests on the latter with different sized topic maps. A very faithful implementation of the merging algorithm can be too resource demanding for a mobile device. Therefore, the challenge was, in addition to implement the algorithm, to optimise with respect to system time and device reasonable shortcuts in the algorithm.

Table of Contents

Table of Figures

1 Introduction

During a rescue operation, there are different organizations involved like police, ambulance and fire fighters. To ensure an efficient operation they need to co-operate across organizational boundaries. This creates the need for information sharing between different systems with different interfaces in a heterogeneous network. Information sharing in a scenario where human life could be at stake, leaves little buffer for latencies or system faults. There is little research in developing systems for the emergency scenario. The communication is very often oral, leading to many repetitions. However, there is also, so far little use of handheld devices by the personnel at the scene. To be able to get information immediately and perhaps across organizations, the use of mobile devices in a mobile ad-hoc network could prove to be a valuable asset.

This thesis is a part of the Ad-Hoc InfoWare project [1] at the University of Oslo. The goal of the project is to enable information sharing between rescue personnel at an emergency scenario by the use of Mobile Ad-hoc Networks. The objective is to communicate across the different domains and organizations involved at the rescue scene by the means of shared ontologies [2]. This is where the challenges lie. One of the modules of the Ad-Hoc InfoWare project is concerned with the knowledge management aspect. This thesis is a part of this module.

To represent ontologies the choice stood between the Resource Description Framework (RDF) [3] and the Topic Maps standard (ISO 13250-2003)[4]. We have chosen the Topic Maps standard. Topic Maps can represent ontologies and is realised as a knowledge layer on top of the different system in different domains. The Topic Map layer will serve as a common ground for all the systems and enable the different laptops, PDAs and mobile phones to share information. The Topic Maps technology has many functions that can contribute to the desired result. Topic Maps is an emerging knowledge representation technology that contains a lot of potential.

So far, there has hardly existed any application of Topic Maps on mobile devices such as mobile phones. A mobile Topic Maps engine was developed recently in Berlin, in connection with the Shark project [5]. The engine was named Mobile Topic Viewer (MTV) and is the basis of the implementation in this thesis. There has been a concurrent master thesis on the same topic. Samuel Vigdal [6] performed benchmark tests by querying different sized topic maps with the use of the MTV engine. The queries were processed within reasonable system time. There is one important aspect missing in the MTV, the process of merging topic maps. The merging of topic maps is considered the core concept of the Topic Maps standard. When merging two arbitrary topic maps, the redundant items are removed and the remaining information is merged. By using a shared ontology the mechanism can connect knowledge across different domains. Topic Maps has many applications that aim to fulfil the idea of “Seamless Knowledge”. To reach this goal the merging functionality is the very heart of a distributed Topic Maps system. To test merging of topic maps on mobile devices, the algorithm was implemented in the MTV.

This project had some uncertainty attached to it since no one had implemented merging of topic maps on mobile devices earlier. The Topic Map data model gives a superficial description of the procedures of merging, so I tried to find implementations of the merging algorithm. It turned out to be very few applications that used merging. In fact, the merging was seldom the focus of the Topic Map applications. I got the impression that merging has not achieved its full potential. Since the Topic Maps standard is primarily targeting usage on resourceful devices, it was important to perform benchmark tests. The question was; could a

mobile device be capable of merging topic maps according to the requirements in the use case scenario. Merging can turn out to be a very resource-demanding algorithm for a mobile device, if one wants to remove every redundant item. Therefore, the challenge was, in addition to implementing the algorithm, to optimise the running of the algorithm.

Problem statement
As introduced, efficient collaboration between rescue personnel from various organizations is a mission critical key element for an effective operation in emergency scenarios. The challenge is how to enable the sharing of information with Topic Maps on resource-weak devices. The solution must be effective and yet this must not lead to loss in details and/or correctness. Will a mobile phone be too resource-weak for the procedure? Rescue personnel depend on simple and quick response from their systems. Will it be useful?

Reader guidance
Chapter 2 presents the theoretical background needed for understanding the requirements and the elaboration of the thesis. This involves an introduction to the Ad-Hoc InfoWare project and a presentation of the emergency use case scenario. The succeeding section elaborates the concepts of the Topic Map standard that are important to be familiar with before implementation of the merging algorithm is described. The concept of ontologies are also explained. The last section introduces RDF, and outlines a short comparison of the two knowledge management technologies, RDF and Topic Maps.

Chapter 3 presents projects that relates to or can contribute to the work done in this thesis.

In Chapter 4, the requirements of the Ad-Hoc InfoWare project is sketched out. An emergency use case scenario with the use of the core concepts of Topic Maps is suggested.

Chapter 5 introduces the concept of merging in a detailed manner. The rules behind the merging algorithm are worked out as a preparation for the technical implementation.

Chapter 6 is about the technical implementation. The basic structures in the MTV are elaborated. Moreover, the merging algorithms are sketched out in pseudocode. To optimise the solution, I have pruned and adjusted the algorithm for its purpose. These solutions and decisions are explained. The trade-offs and shortcomings in the solution is also worked out. Finally yet importantly, the benchmark tests are introduced and commented.

Chapter 7 discusses and concludes the findings and results in the thesis. The discussion is based on the implementation and the findings in the master thesis. Further work is sketched out to improve the algorithm and continue the work.

When you encounter this symbol in front of a section, this indicates a high difficulty level. These sections can be skipped if desired, and this will not influence the general understanding of the rest of the thesis. It may be, in some cases, an advantage to skip these sections and concentrate on the rest.

I will write Topic Map, with capital letters when I refer to the standard and technology. A topic map e.g. the elaborated representation expressed syntactically is written in small caps.

2 Background

2.1 The Ad-hoc InfoWare Project

The background of this thesis is the work and initiative of the Ad-Hoc InfoWare project. This chapter is based on [1] if no other reference is given. The Ad-Hoc InfoWare project works with and explores the development of middleware services to facilitate information sharing in Mobile Ad-hoc Networks at rescue scenes. The goal is to simplify application development by developing middleware services. The project identifies six middleware concerns, and the knowledge manager is one of them. This chapter will give an overview of the project and a more in depth description of the needs and concerns of the knowledge manager.

Mobile Ad-hoc Networks
The Mobile Ad-hoc Network (MANETs) is a network solution that can be of great use, especially in the emergency scenario. The end-user devices such as PDAs, mobile phones and network interfaces that are available for these devices have made the use of ad-hoc networks possible[7]. In the rescue service there is a large need for information sharing amongst different systems in such networks. Since this is a mission-critical service it has a low tolerance for package loss and of incorrectness of the information. At emergency scenes where several rescue teams work together and has the need for correct information immediately, and the ability to share it, this is an important issue. The need for sharing, receiving and distributing data can be between different organizations like paramedics, firefighters and police officers. In these scenarios, there will be teams that have different systems and different interfaces. This is where the challenge lies. To enable application development there is a need for middleware that can support and facilitate the sharing by developing a common knowledge module.

The knowledge manager
The next sections are based on [1, 8]. Figures 1 and 2 are also taken from [1]. The middleware service will serve as a common ground for all the different systems and wireless network interfaces and facilitate information sharing in the MANET. This includes both inter-and intra-organizational information sharing. This means that there is a need for a translation between the ontologies and metadata standards used in the organizations, as well as a standard language for information exchange. Since most of the relevant ontology and metadata standards are represented in XML, this language has been chosen as the international standard for message exchange in the health sector.

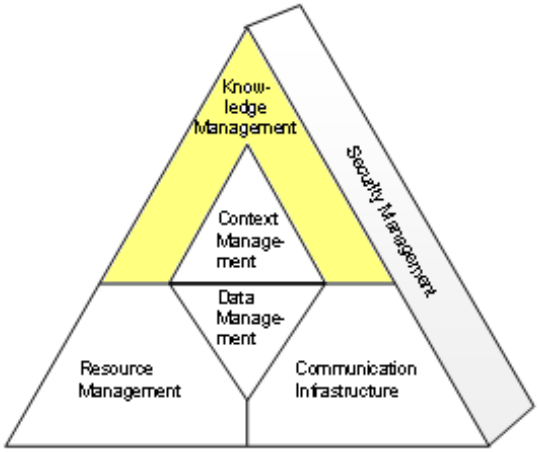


Figure 1Knowledge manager concerns

For the knowledge manager, the idea is to implement ontologies to promote shared vocabularies. These vocabularies should be machine-processable and be explored by the aid of browsers, query engines, or reasoning engines. This will make the information available and global. Some of the challenges are getting the right information at the right time, with the right format and granularity. Also filtering the information is essential. Overflow of information is as bad as getting no information and must be avoided. The ability to limit the scope of a search to fit a profile and to limit the validity of an information item to a certain context is a part of this problem.

Knowledge management requirements

The first and foremost objective of the knowledge management is to support distribution, sharing and interpretation of ontologies. The next is to be able to browse and query ontologies and ontology content. These objectives created a need for distributed knowledge base functionality and a global view of what is available in the network. Again, this resulted in a set of issues; different domains and organizations, infoglut; unavailability of information, information query and retrieval services and information exchange. Below follows a more detailed list of the main concerns which the knowledge manager will have to address:

- Understanding across domains and organizations
- Global view of available knowledge through the use of ontologies and shared vocabularies
- Inter-and intra organizational information exchange
- Avoid of information overflow
- Content filtering and personalization

Requirements for the ontology language, address issues like; expressiveness, completeness, correctness and efficiency, and interoperability with other relevant standards.

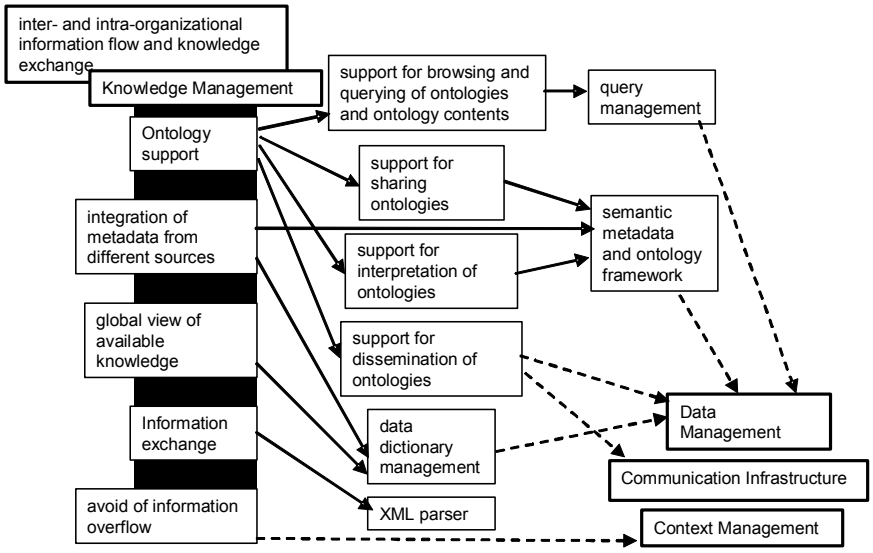


Figure 2 Requirements of the Knowledge Manager

The emergency process
The project identifies six phases of an emergency process.

Phase 1- *A priori.* This is the phase before the accident. The different organizations will in cooperation with the authorities exchange information on data format and make agreements on procedures and working methods.

Phase 2- *Briefing***.** The step after an accident has occurred involves information gathering and some preliminary decision making as to procedures.

Phase 3– *Bootstrapping.* The bootstrapping of the network and registration of nodes take place at the emergency scene. Every security aspect is taken care of.

Phase 4– *Running of the network.* This is where the knowledge management module steps in and solves the information sharing issues and enables intra-and inter-organizational flow e.g. by handling ontologies.

Phase 5 – *Closing of the network.* The application adapts to the closing of the network by acting on the received information about degradation of the capabilities and resources of the network.

Phase 6– *Post processing*. Depending on the nature of the application, it could have e.g. gathered data to be used later in statistics.

2.2 The Emergency Use Case

This section is based on discussions and dialogues with Samuel Vigdal.
The elaboration of a use case scenario has been very helpful in the process of understanding the situations an emergency crew could encounter at work. It is an attempt to reveal parts of the communication and co-operation that could be made more effective or has a potential for improvement. It has also been a very important and a valuable starting point when

considering Topic Maps as a suitable knowledge representation standard for information sharing. It is impossible to predict an emergency scene in detail, since every accident is unique. Decisions must be taken at the scene in accordance with the nature of the accident and its course. Despite this fact, some elaborated regulations and guidelines should be followed to ensure a most effective rescue operation. The objective of this Chapter is to exemplify an imaginary emergency scene based on some guidelines that exist in the rescue service in Norway today.

2.2.1 Emergencies

Today there is an increasing focus on terror threats, catastrophes and other possible disasters caused by nature or man. After the 11th of September and the ongoing disagreements between religious fanatic groups especially in the Midwest, the need for and interest in improving the emergency preparedness has increased. The augmentation in nature disasters like tsunamis and hurricanes has also attracted attention to the handling of such catastrophes. Since 1984 NTNU (Norges Teknisk-Naturvitenskapelige Universitet) and TrygVesta has arranged a seminar called Sikkerhetsdagene *(the Safety Days)* [9] in Norway. This started out due to a large amount of big accidents in the mid 80s, which required rethinking in the safety field. The globalisation and the growing IT industry have forced us to be updated and follow the continuing changing world. In order to meet with the potentially large accidents and improve the effectiveness and procedures at place, there are numerous projects under development. The Directory for Society Safety and preparedness in Norway is developing a project called NARRE (Nasjonalt register for rednings-og beredskapsressurser) [10]. This is aimed at creating a national net-based register containing all the rescue resources available. Now a pre project is being tested in Hordaland.

What is an emergency/crisis?
According to the Directory for Society Safety and preparedness in Norway [11], an emergency or crisis situation can be described by the following characteristics:

- The emergency arrives surprisingly
- The lack of control
- Vital interests is at stake
- Many actors
- Time is short
- A breakdown in the regular decision-process
- Focus on short term solutions
- Uncertainty
- Lack of information
- Intense interest and follow-up from outsiders

2.2.2 The Imagined Scenario

This section is based on [12] [13] where nothing else is given.

Example scenario: two trains have had a head-on accident inside a tunnel. The two trains consist of six carriages each and there are approximately 10-15 persons inside each carriage. There has been a fire in some of the wagons. The only information the rescue personnel has received is that there has been a big train-accident inside a tunnel, and the number of people involved can range from 15 to 100 persons. The number of people involved and the fact that it

is a train accident indicates a need for a large number of emergency personnel and coordination between the emergency groups. The responsible rescue organisation for that part of the country is contacted through 113. The fire department, police force and ambulance are informed.

To make sure the operation is as effective as possible and as well driven as possible, it is important that the leaders follow instructions. Today, there has been developed many procedures and rescue plans for handling this type of accidents. Below follows, a sketched out possible sequence as it could have happened today. The example is simplified and it does not include all the roles and stages in an emergency scenario.

The normal procedure is as follows; the head of the police is appointed Emergency Officer (EO) at the scene of the accident. Further, a head of fire and a leader of order are elected. Within health, there is an operative leader for health and a health leader. The EO will coordinate the efforts of the leaders and support them in performing their tasks, namely save lives and prevent further damage. She/he will also provide the necessary resources through the local health service. In addition, the EO will set up a place of command, where she/he will conduct the efforts through the leaders.

The health leader is medical responsible on the scene, this position is not always appointed. The operative leader in health reports to the health leader. The person shall organize, lead and coordinate the operative sanitary unit, and in addition organize evacuation in cooperation with the EO and the AMK (Acute Medical Communication) central. Both these roles must act in accordance with the EO when it comes to coordinating tasks in the rescue area and when in need of resources outside the public health service (helicopters, light etc). They must also be in touch with the AMK central for requests concerning resources from the public health service, and the distribution of the patients to the next level in the treatment chain.

During the operation, the different leaders are in charge of their groups, and the internal information and coordination. However, the operational leader is responsible of the co-ordination between the groups and in general makes sure everything works smoothly. The group leaders are dependent on the operational leader when it comes to updates on the scene. All of these tasks are both important and crucial tasks.

Today, all information is being communicated through radio, e.g. by using Terrestrial Trunked Radio (TETRA)[14]. TETRA is a global and open standard for professional mobile radio communications, defined by the European Telecommunications Standard Institute (ETSI).The EO has to repeat information to personnel and update newcomers. These many redundant and repetitive tasks could be automated. In addition, the information received on the radio cannot be filtered. This can result in many repeated messages, which is waste of time. This time could be spent on other tasks. Data from the scene, e.g. patient status/injury is registered in the ambulance. The information is at that moment communicated verbally. At the scene, there is no electronic marking of the most critical injured persons. This could lead to the fact that the persons who need help immediately will have to wait. This is the general idea of the course of an emergency scene. Several aspects can benefit from the introduction of knowledge management.

2.3 The Topic Maps Paradigm

Today there is a constant and increasing problem with information overload and infoglut. The Web is flooded with new pages every day and searching on the Internet often leaves us with too many choices or the wrong results. These issues can also be reflected in organizations and

companies. They are also struggling to maintain an overview and being able to navigate in their assets and information resources. Important documents can be lost or connections between related information are perhaps never captured. Large companies with many departments can easily loose important communication between internal systems. This leads to a connectionless environment, where resources are never reused across sections, and organizational memory is partitioned instead of gathered. There are many scenarios that demonstrate the need for a different knowledge management approach. In a distributed environment, there is need for a common platform and a system for connecting the distributed information together. We already have some solutions to the problems in form of attaching metadata to resources or using keywords. However, these solutions create new problems; metadata cannot create relations or context to other resources. Keywords are added in a very subjective manner with no control of the vocabulary used. The solution is Topic Maps! Topic Maps is a knowledge management standard that enables amongst many other functions; subject-based indexing, navigation and filtering.

The history of Topic Maps
A discussion started at ACM Hypertext '91 in San Antonio [15]. The idea of being able to merge (book) indexes was the starting point of the development of a long discussion which eventually developed into Topic Maps. Some of the most significant founders were Steve Newcomb and Michel Biezunski. The standard had to be powerful and flexible on the one hand and have sufficient well-defined semantics on the other hand.

In January 2000 the International Organization for Standardization (ISO) approved and published ISO/IEC 13250:2000 Topic Maps. The standard also defined syntax for Topic Maps. The syntax was SGML DTD, and used the ISO 10744 HyTime standard for linking and addressing, and so the syntax is known as HyTM (short for HyTime Topic Maps). HyTM had some shortcomings; among others it did not use URI (Uniform Resource Identifier) for addressing resources. This did not integrate very well on the Web. To resolve the lacking functionalities the TopicMaps.org organization took over further development. Since XML was a widespread language for Web and used URI for addressing the new syntax was based on XML. XTM (XML Topic Maps) was approved by ISO in October 2001. A query language (TMQL) and a constraint language (TMCL) were started some years later. They are still under development but will soon be stable. These projects are under the supervision of ISO. The latest version of the Topic Maps standard came out in 2002 [16].

The present
The Topic Maps standard has a passionate and active community. Norway and Germany are amongst the active countries with very many motivated developers. The Norwegian company; Ontopia (http://www.ontopia.net) is one of the leading developers of Topic Maps-based systems in the world. They have partners all over the world amongst them are Neofonie (http://www.neofonie.de) in Germany. There is a Topic Maps conference in Germany each year, called TMRA. On this conference, every new addition to the Topic Maps standard is introduced, in addition to applications and solutions. In Norway there is a similar conference called Emnekart (Topic Maps). Topicmaps.Org is an active and independent consortium of parties interested in developing the applicability of the Topic Maps paradigm to the World Wide Web, by leveraging the XML family of specifications as required.

The future
The vision of Topic Maps is to obtain “Seamless Knowledge”. This is not to be confused with the Semantic Web, which is a concurrent vision in the knowledge management community. Seamless knowledge is an expression that describes a knowledge flow across organizational boundaries and systems. The idea is to connect distributed information without seams.

However, Topic Maps is now also considered a part of the Semantic Web. This is a common framework for sharing and reusing data on the web, across applications, communities and organizational boundaries. It is based upon the Resource Description Framework (RDF) and led by W3C, together with industrial partners. The effort focuses on the Web and enabling people to be independent on the applications they are using. Very shortly described, it should be possible to e.g. read Word documents in Outlook and your banknotes in Word, [17]. There are ongoing projects for mapping RDF documents to Topic Maps and vice versa. [18].

Topic Maps is originally an ontology-based technology. The standard can express any ontology and inhabits many other functions and mechanisms. Before I present the Topic Maps paradigm, an introduction to ontologies is in place. When you understand the concept of ontologies, it can be easier to grasp the essence of the Topic Maps standard.

2.3.1 Ontologies

Ontology is originally a concept in philosophy, describing the study of being or existing [2]. Ontology can be considered a more complex version of taxonomies. A well-known taxonomy is the classification of the species by Carl von Linné, performed over 200 years ago [19]. In the computer world, ontologies have a bit different definition, although the idea is the same.

In computer science, an ontology is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them.

We can either describe a domain by using a domain specific ontology or describe a more general ontology by creating a *common* (upper) ontology. A *common* ontology is a universal ontology that can be used across many domains. It serves as a common definition on a higher level .A domain can be e.g. the *rescue service*. The ontology consists of set of axioms to formally define concepts and relations that shall exist in the mentioned domain. This means that the domain could contain concepts like, *injury*, *vehicle*, *accident* and the different *resources* available. The ontology can also assert information about individuals in the domain, in other words; add attributes to them [20].

The ontology will work as a common vocabulary, to enable a correct communication. If organizations from Norway and Finland where to co-operate, they would need to establish a common vocabulary in a common language; English. This would ensure a common understanding of the information. They could decide to use the word *accident*, instead of *incident, ulykke*, *rescue scene*, *emergency scene* or other names. The ontology can be used to control the vocabulary by defining which expression shall be used. These definitions make up a common knowledge layer for communication across different systems. It avoids misunderstandings and enables contact beyond language and organizational context.

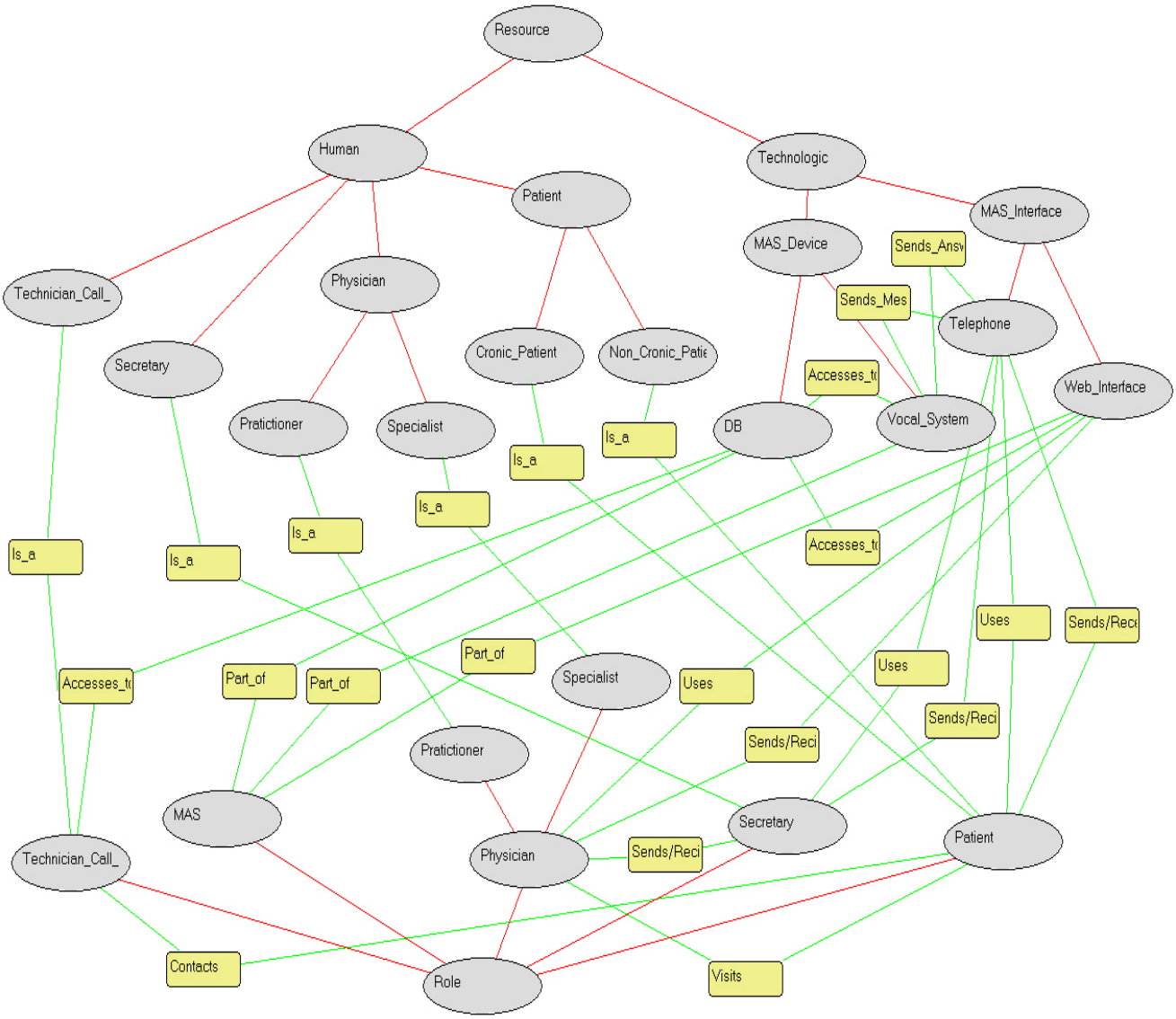


Figure 3 Ontology

This lists [21] the basic advantages the elaboration of an ontology can offer;

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational knowledge
- To analyze domain knowledge

The development of the ontology can be very challenging. To create a domain specific ontology, the creators need a very thorough understanding of the big picture and the details in the given domain. A metamodel can also be considered as an ontology. There are ongoing projects that develop ontologies for use in rescue operations.

Ontologies have been elaborated to support development of knowledge management systems in the emergency field. One example is the Emergency GIS based project [22], where different ontologies are developed to be used in an application for the rescue operations.

Figure 3 is an excerpt of an ontology from a project that researches on a Web-based Health Service Flow Management System[23]. They have developed amongst many, this ontology as a basis for their flow charts. This could be an excerpt of a domain specific ontology of the ambulance organization. The super-and subclasses expresses the hierarchy; in addition the relations between the classes/subclasses are very well expressed by naming the relations. To express the ontology syntactically the Web Ontology Language (OWL) is used [24].

The ontology consists of *types* of things that consist in the medical world. There are no instances of *patient* or *physician* in the model. This is because the model is supposed to describe the domain, not the actual contents. Topic Maps can do both.

Now you are ready for the world of topic mapping.

2.3.2 The Standard

To explain the relationship between ontologies and Topic Maps I will draw a parallel with other technologies. Ontology in Topic Maps corresponds to the set of element types and attributes in XML or the set of tables and columns in Rational Database Management Systems (RDBMS). In other words; the ontology determines what you can assert or *say* in the topic map. When starting topic mapping, the first step is to create the ontology [25, 26]. The next step is to populate the topic map with *instances* of the ontology. If *physician* is a part of the ontology, an instance of this item could be *Mary Johnsen* (presupposed that she is a physician). The ontology supplies super-types, while the topic map layer consists of instances of the super-types. The topic map contains both the ontology and the instances. The Topic Map standard do not usually make a division between the two layers, but rather view every item in the topic map as topics.

> *"An International Standard for subject-based organization of information and knowledge management" [27]*

Topic Maps is a knowledge representation standard that is aimed at solving the *findability* problem by organizing knowledge in a subject-based manner. The objective of the standard is to connect *disconnected* information. The usage of topic maps as knowledge management in distributed systems can enable communication across different systems.

Topic maps consist of topics and the relationships between them (associations). The topics can link to information resources outside the topic map. These resources can be contents in databases or servers, XML documents, files, images, everything that can be presented electronically. The information resources are *not* a part of the topic map. The topic that links to an information resource represents the resource and says something about the contents of the resource. The topic adds *metadata* to the resource. The associations are used to creates relations between the topics and hence create relations between the information resources. Associations are the glue of the topic map and they are what enable browsing.

According to Figure 4, the topic map exists in a separate knowledge layer. In the figure, the information resources are located on a server. The advantage of this two-layered structure is

loose coupling between the external information resources and the topics that represent them. This makes it unnecessary to make changes in the persistence layer. Different topic maps can be used to represent the persisted information in several ways.

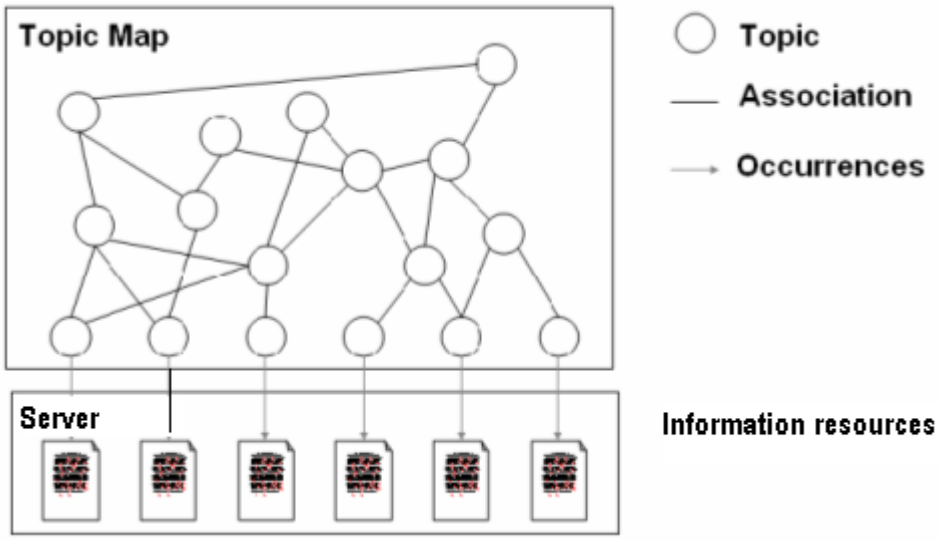


Figure 4 The Topic Map model

Everything in a topic map is a topic or a relationship between topics expressed as an association. There are several applications of Topic Maps and many different approaches to the standard. Topic Maps were originally aimed at handling indexes, thesauri, glossaries and table of contents [18]. To explain the purpose and origin of Topic Maps in a more detailed manner, an introduction and comparison to the library science of indexing is due.

Indexing

A book index gives a map of the contents of the book, based on topics. Without an index we would have to read the whole book to find a certain topic. In the example below *Rescue operation*, *Fire leader* and *Injury* are topics and the page numbers are addresses (occurrences) indicating where the given topic is to be found.

Emergency, See Rescue operation
Accident, 68
    Espa train accident, 45
Accident scene, 48
Emergency personnel, 24
    Emergency officer, 20
    Fire leader, 6, 15
    Police man, 10
Rescue operation, 24, 47, 64
Emergency officer, 15, *See also Emergency personnel*
Injury, 27, 52
Espa train accident, 45
Location, *See Accident scene*

Figure 5 Book index

This simple index example demonstrates the basic concepts of Topic Maps. The index is organized by topics and the occurrences of the topics are attached by using an addressing scheme, in this case; page numbers. Instead of page numbers Topic Maps use URI (Uniform Reference Identifier) to address the information resources [25, 26]. *See* and *See also* are used as associations to build relationships between the topics. The topics are connected and put in a context.

To avoid ambiguity with synonyms the *See* expression is utilized. Since *accident scene* is the preferred term before location, the *See* expression guides the user to the correct term. The *Emergency personnel* index illustrates the sub/super class hierarchy by listing the sub types below. The *See also* expression is used to indicate a relationship between topics/index. Figure 6 illustrates the mapping between a book index and the Topic Maps structure.

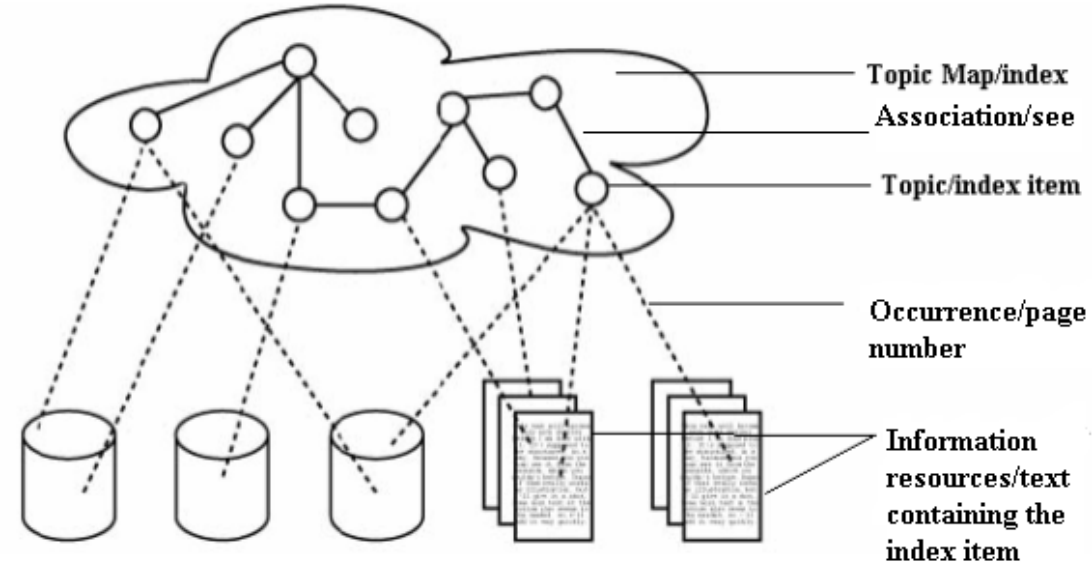


Figure 6 Topic Maps and indexes

As mentioned, Topic Maps is an ontology-based technology and therefore supports ontologies. The ontology layer contains the topic types i.e. the domain specific ontology e.g. *person* and *accident*. The lower layer consists of topic type instances. Figure 7 illustrates this.

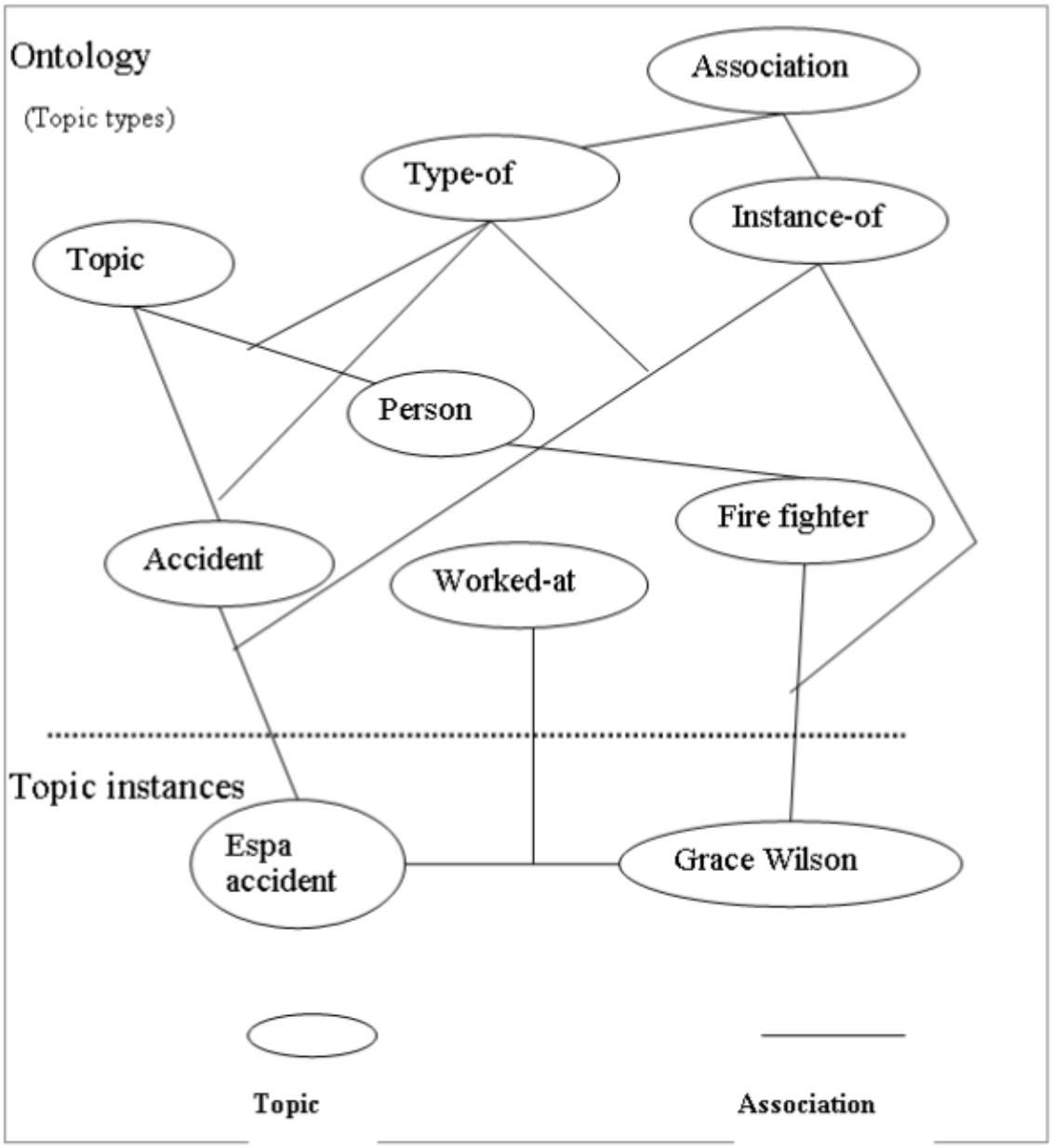


Figure 7 The topic map

Both layers are a part of the topic map. As you can see in the example above, everything is topics, except the associations, which are marked as lines between the topics. The topic named *association,* is *not* an association, it is used as a super-class for the *type-of* and *instance-of* topics.

Family of standards and languages
Topic Maps consists of a family of standards although not every specification is approved yet. Figure 8 expresses the family of standards as a topic map [28].

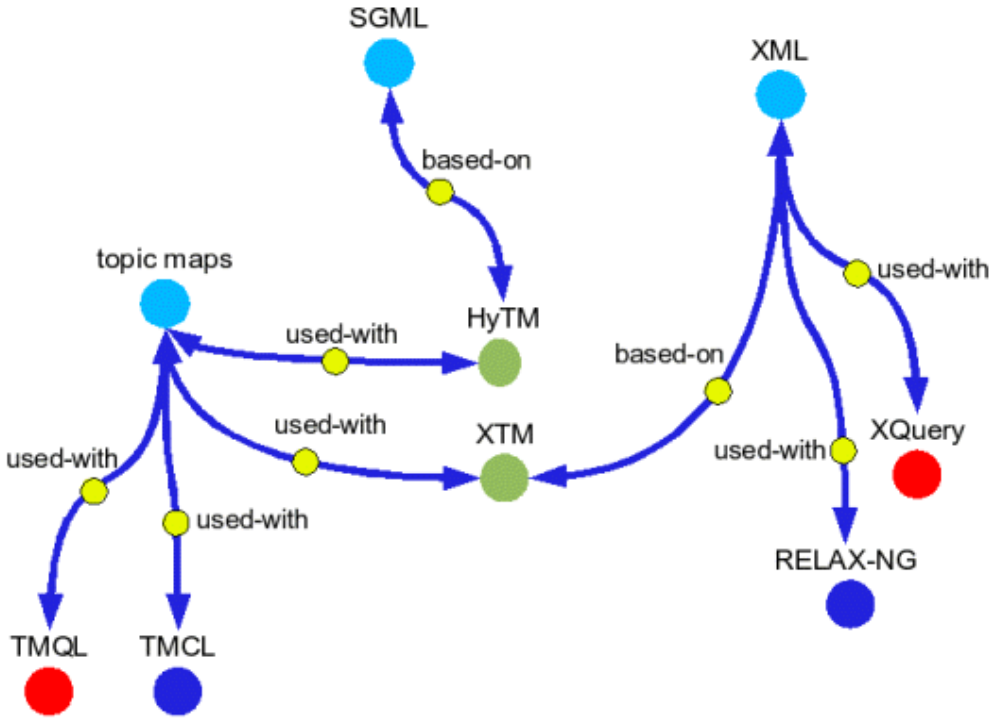


Figure 8 The family of standards

XTM
Topic Maps has a standard XML-based interchange syntax called XML Topic Maps (XTM) [29]. There is a final draft 2.0 specification out for review, which is known to be stable. Since the Mobile Topic Viewer is based on the XTM 1.0 version, the technical implementation is also based on this. To avoid confusing the examples in this chapter will be based on the XTM 1.0 version. Some of the changes in the latest review concern a simpler naming, but I will not go into this.

HyTM, LTM and AsTMa=
Other syntaxes exist, e.g. Hytm, LTM, and AsTMa=. Hytm was one the first notations used and is more or less deprecated and substituted by the newer ones [30]. The Linear Topic Maps Notation(LTM) is a compact and simple notation for use in small demonstration purposes [31]. AsTMa= is a part of the AsTMa language family [32]. This notation is targeted for human authoring and must, like LTM, be converted to XTM for use on the web[32]. It is less verbose then XTM and is suppose to be easier to write topic maps. The AsTMa project has designed a family of languages which is aimed at updating, constraining, authoring and querying topic maps. The software is free and Python-based

TMQL and TMCL
Since Topic Maps are based on a data model, the topic maps can be queried like a database. A query language, called Topic Map Query Language (TMQL) [33], is currently under development by WG3 . To be able to query a topic maps system across applications will

enhance portability. Meanwhile, Ontopia has developed TOLOG for performing queries. Other query languages are TOMA and TMPath. There is also a schema language, the Ontopia Schema Language (OSL) [34], for expressing rules and constraints in the topic map. This effort has been used while ISO finishes the, to be called ISO 19756 Topic Maps Constraint Language, or TMCL.

2.3.3 The Data Model

The first official draft of the Topic Map Data Model (TMDM) [35] came this summer. I am referring to the TMDM in this section, where nothing else is given. The XTM 1.0 specification is used in the examples since this is the version used in the implementation in the Mobile Topic Viewer. There are some syntax differences between the XTM 2.0 version and the first version, but this does not affect the data model profound.

> *A topic map is a set of topics and associations. Its purpose is to convey information about subjects through statements about topics representing those subjects. The topic map itself has no meaning or significance beyond its use as a container for the information about those subjects.*

This quotation demonstrates that the topic map itself is merely a container and refers to the actual document. The topic map will not be elaborated, but the components it contains will be.

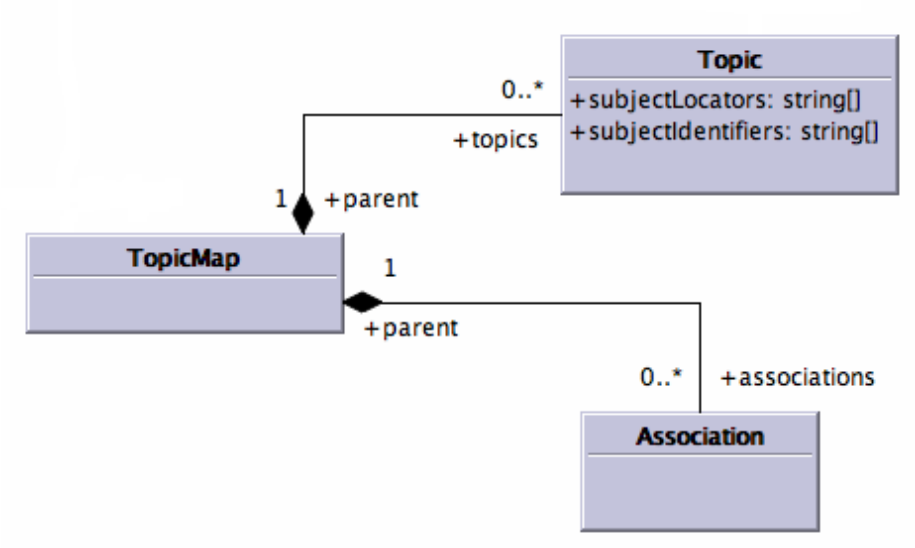


Figure 9 Topic map model

Figure 9 illustrates the relationship between the topic, the association and the topic map itself. A topic map can have zero or many topics and/or associations and the topic item and the association item can and shall have *one* and only one topic map as parent.

Connecting it all together

The relationships between topics are expressed by using the association item. The association is given an intuitive name which is used to capture the type of relationship between the topics. The association is directionless. In Figure 10, the association is used to connect the topics together and give the relations semantics. The Topic Maps standard often refers to *the TAO of Topic Maps* when summarizing the basic concepts. This stands for *topics*, *associations* and *occurrences*. To give the reader a profound insight into the standard I will elaborate the TAO of topic maps. The first letter in the abbreviation stands for the *topic* in Topic Maps.

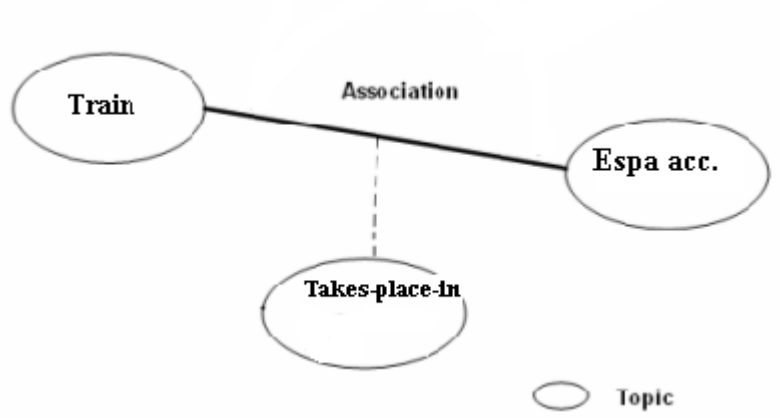


Figure 10 Excerpt topic map

2.3.3.1 The Topic

To be able to index or navigate in information resources in a subject-based manner, we must have topics in the topic map. The topics represent subjects in the domain of question, e.g. in the *ambulance-world.* Examples of subjects could be; *damage*, *injury*, *accident, nurse* etc. These subjects must be represented [1]by a topic. This means that the topic serves as a proxy for the subject of matter. The idea '*accident'* is represented by the topic: *accident*. This has many relations to philosophy and the allegory of the cave by Plato.

In other words; the topic can be anything, either abstract or concrete. As described in the topic map data model:

> *A subject can be anything whatsoever, regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever. In particular, it is anything about which the creator of a topic map chooses to discourse.*

An information resource e.g. an article about *police men* will create a need for representing the idea *police man/men*. As Figure 11 illustrates, *Policeman* will be represented by creating a *policeman* topic.

[1] Also referred to as reification in the data model, translated to "thingification".

Figure 11 Representing a subject from the real world

Figure 12 illustrates the data model of the topic item. The model demonstrates how the topic is involved with the other items in the topic map. Occurrences and topic names are properties of the topic. A topic can be a part of an association role. The association role is a part of the association which will be explained in next section.

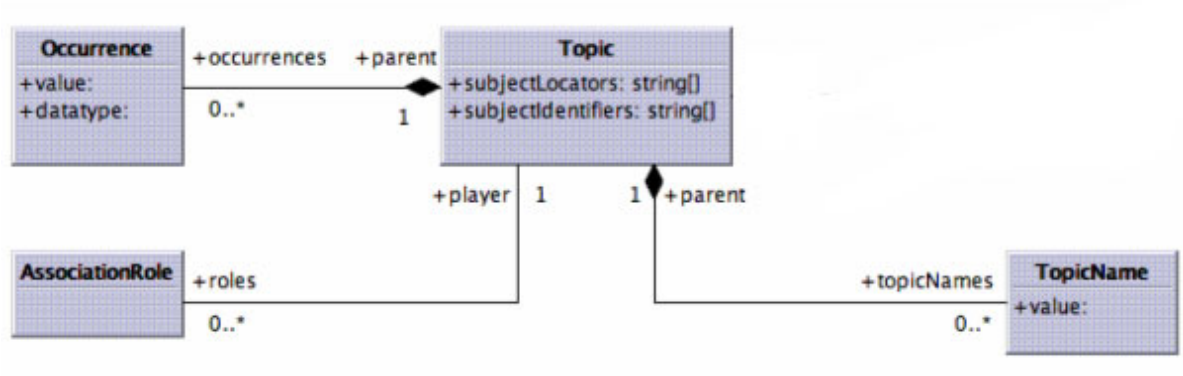


Figure 12 The topic item

The topic's properties

The topic has many properties. The properties are sorted in taxonomy to express the hierarchy of the topic item.

- [subject identity] (*must* contain at least *one* or both)
  - subject locator(s) * or PSI(s) *
  - Item identifier(s) *
- [name(s)]*
  - variant(s) *
  - scope
- [occurrence(s)]*
  - scope
  - type
- [roles played in associations]*

- *[type(s)]*[2]

The items marked with a * can be singular or multiple. A topic can have multiple names, where each name can have multiple variants. The topic type property is expressed by using *instance-of* associations according to Figure 14. I will refer to [type] as a property of a topic for simplicity.

Deserialization and serialization

A topic map can be represented by using XTM syntax, to enable the Topic Map engine to process and query the topic map, the topic map must be deserialized. Deserialization is the process of building an instance of an implementation's internal representation of the data model from an instance of topic map syntax. During this deserialization, a unique string must be provided as input. To enable unique naming, it is recommended to use the absolute path to the location of the topic map document. This is referenced as the *locator* of the topic map. A locator is given by the application or the user. Figure 13 illustrates the deserialization and the serialization process.

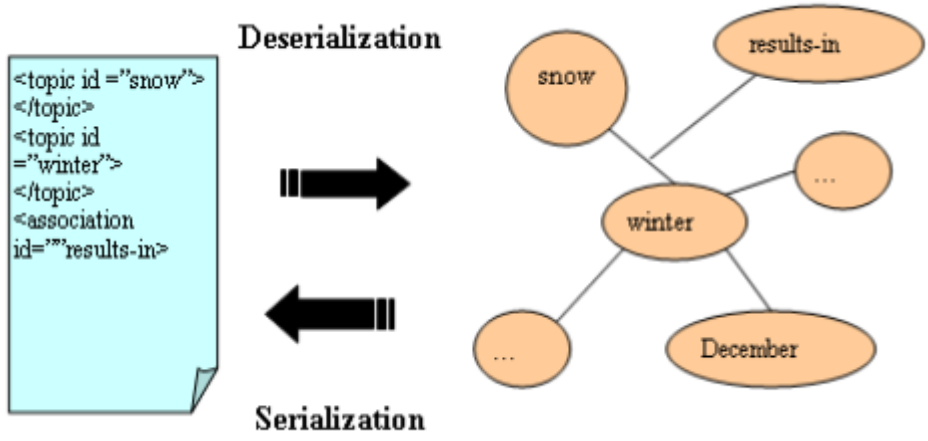


Figure 13 Deserialization and serialization

Item identifier

The item identifier is the concatenation of the id attribute of the topic in the XTM document and the locator of the topic map. In the example below, the id of the topic is; *espa_train_accident*.
Topic map locator: http://www.example.org/tm/my-tm + # +

```
Topic id (XTM) :<topic id="espa_train_accident">  ... </topic> =
```

Item identifier: http://www.example.org/tm/my-tm #espa_train_accident

After a merging process, a topic map item can have several item identifiers. When serializing the merged topic map (writing the topic map constructs back to XTM format) the item identifier information is lost according to XTM 1.0. In the newer version, the item identifier can be expressed on the same level as subject locators and/or published subject identifiers. By using the XTM 1.0 syntax, information can get lost when serializing a merged topic map.

[2] Type is not a property of a topic according to the data model, it is expressed by using associations

Topic names and scopes
In addition to its identity, a topic should have a name but this it is not required. In the XTM 1.0 version the name is denominated as *base name*. I will use *topic name* for simplicity. This name should indicate the nature of the topic. The former example will then be named in the following manner.

```
<topic id="espa_train_accident">
  <baseName>
        <baseNameString>Espa train accident<baseNameString>
</baseName>
 </topic>
```

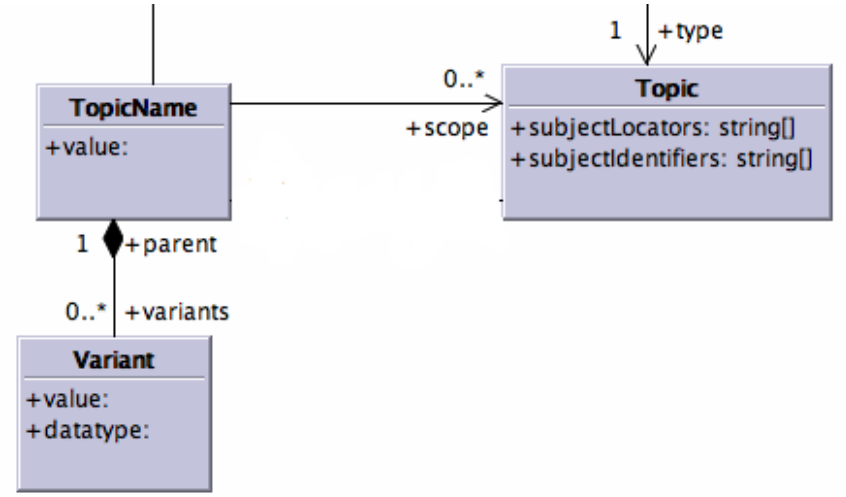


Figure 14 Topic name

The data model recommends that the context, in which the topic name is valid, should be given. This is to avoid ambiguity or misconceptions. The concept of scoping is what makes this possible in Topic Maps. A scope is the context in which a name is valid. A topic can have several names with different scopes. There can only be one topic name per scope. A scope could for instance be *the language* the topic name is expressed in, which gives the following syntax,

Variants
In addition to several topic names in different scopes, a topic name can contain several *variants*. There can be multiple variants inside a topic name. If there are many variants inside one topic name, the variants should be scoped, to differentiate them. A variant can be scoped in the same manner as topic name.

```
<topic id="espa_train_accident">
     <baseName>
        <scope>
            <topicRef xlink href: =”#english”/>
        </scope>
        <baseNameString>The Espa train accident<baseNameString>
    </baseName>
    <baseName>
        <scope>
            <topicRef xlink href: =”#norsk”/>
        </scope>
        <baseNameString>Espa tog ulykke<baseNameString>
    </baseName>
 </topic>
```

```
<topic id="espa_train_accident">
    <baseName>
        <baseNameString>The Espa train accident <baseNameString>
        <! -- form for sorting (sort name) -->
        <variant>
            <scope>
                <topicRef xlink: href="#sort"/>
            </scope>
            <variantName>
                <resourceData>Espa, train accident</resourceData>
            </variantName>
       </variant>
    </baseName>
</topic>
<topic id="sort">...</topic>
```

The example illustrates how one can provide an alternative name for sorting purposes[3].

Topic types

Each topic can be categorized by defining the class the topic belongs to. A topic can belong to zero or many classes. Topics are used for representing classes; such topics are called topic *types* to indicate the range of use. However, they act and look like a normal topic. The topic *The Espa train accident* can be typed as *accident*. Figure 15 demonstrates a simple excerpt of a type hierarchy in the emergency domain. There are three topic type instances in the figure; the *Espa accident*, *Train456,* and *Grace Wils*. A more detailed description of typing is elaborated in 2.3.5.4.

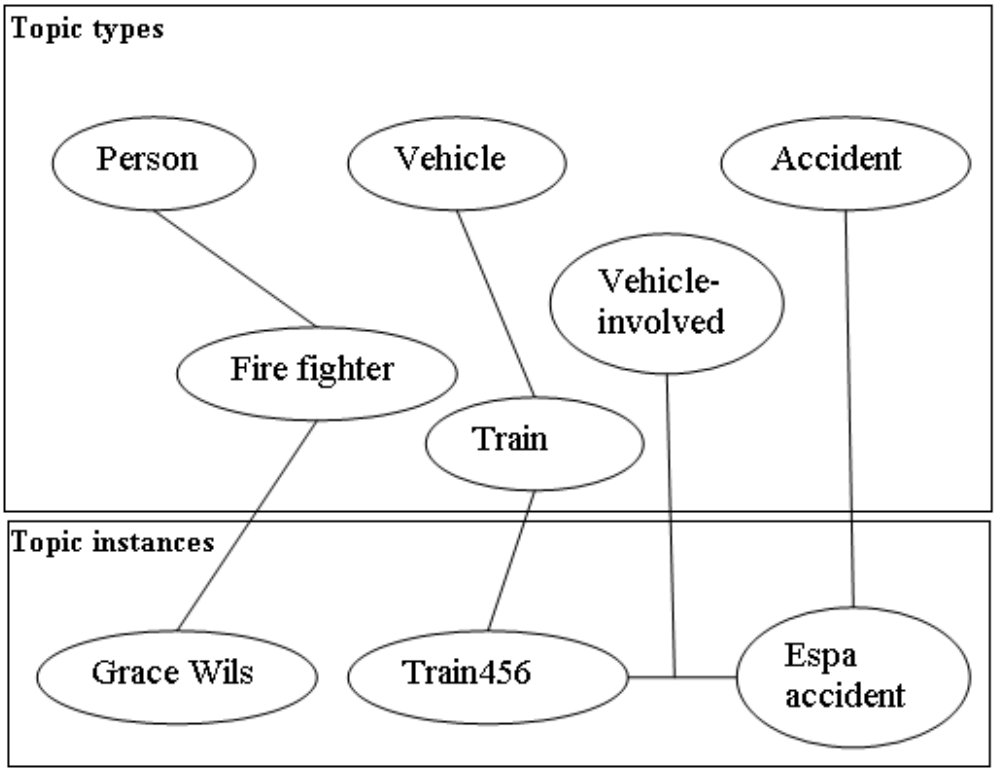


Figure 15 Topic types and instances

[3] The <parameter> element is replaced with the <scope> element for readability.

Both the topic types and the topic instances are parts of the topic map. The information resources reside in the information layer. In the example below, the XTM syntax demonstrates that the *Espa train accident* is of type *accident*.

```
<topic id="espa_train_accident">
    <!—is of type accident - ->
    <instanceOf>
        <topicRef xlink: href="#accident"/>
    </instanceOf>
 </topic>
```

Establishing the subject identity

The objective with topic maps is to have a one-to-one relationship between the topic and the subject it represents. This ensures that all knowledge about a subject can be retrieved through *one* topic. When two topic maps merge together it is important that there is only one topic per subject, see Figure 16. By establishing a subject identity for each topic, it is possible to define the equality of two topics. this can ensure a correct merging[36].

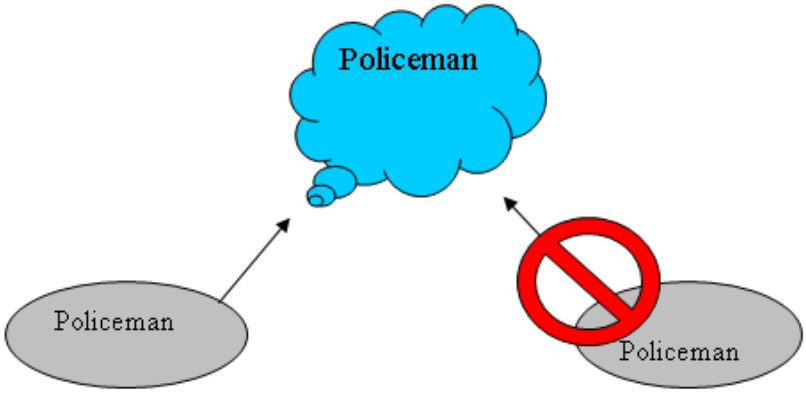


Figure 16 One subject per topic

A subject can be ambiguous. The subject *apple* can correspond to the fruit *apple* or the company *Apple*. A name can have synonyms, homonyms and be expressed in different languages. It is not reliable enough to use the topic name to establish this. This demonstrates the need to define the subject by using another approach. The solution is the usage of *identifiers*[25].

There are two types of identifiers based on the type of subject. The subject can be either an addressable subject or a non-addressable subject. This leaves room for two different identification mechanisms; subject locators and Published Subject Identifiers (PSIs).

Subject Locators

An electronic addressable subject can be e.g. a web site, an article or an image. In other words; it must be something that can be addressed by using the URI addressing schema. When a topic represents an addressable subject, the subject *is* the identity. The subject address is expressed by a locator, called *subject locator*. As the name indicates, it is the location of the subject. According to the ISO standard a topic can have multiple subject locators. Examples of this could be web addresses that are redirected to the same web page. Another example is the UNIX file system, which allows the creation of symbolic links that point to files or directories. This is a way of creating shortcuts to files with long paths. These scenarios allow

several subject locators to the same subject. However, these cases are quite rare and the general use case is to have one subject locator.

Figure 17 illustrates the usage of subject locators. The document on rescue standards is represented by the *Rescue_standard* topic. The URI http://www.rescue.org/rescuestandard.pdf gives the location of the addressable subject.

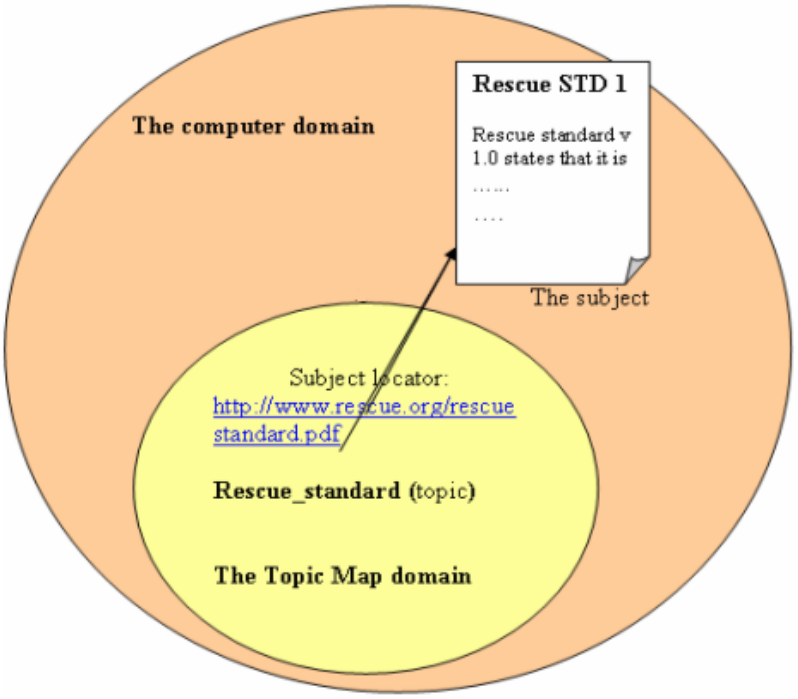


Figure 17 Subject locator

The example below demonstrates a topic representing an addressable subject and establishes the identity with the <subjectIdentity> syntax. A good rule of thumb is to think that subject locators locate the subject. They are addresses to the actual subject.

```
<topic id="rescue_standard">
    <subjectIdentity>
      <!—the subject locator - ->
      <resourceRef xlink:href="http://www.rescue.org/rescuestandard.pdf"/>
    </subjectIdentity>
</topic>
```

PSIs

When the subject is a non-addressable subject, it must be identified in another way. This is done by using a *subject indicator*; a written description of the given subject. This description is referenced by using an address, expressed as an URI. The address is referred to as a

Published Subject Identifier (PSI). The PSI gives a unique URI to the unique description (the subject indicator) of the subject. The subject *accident* does not represent an addressable subject; it is an abstract and therefore could be defined by a PSI, according to Figure 18.

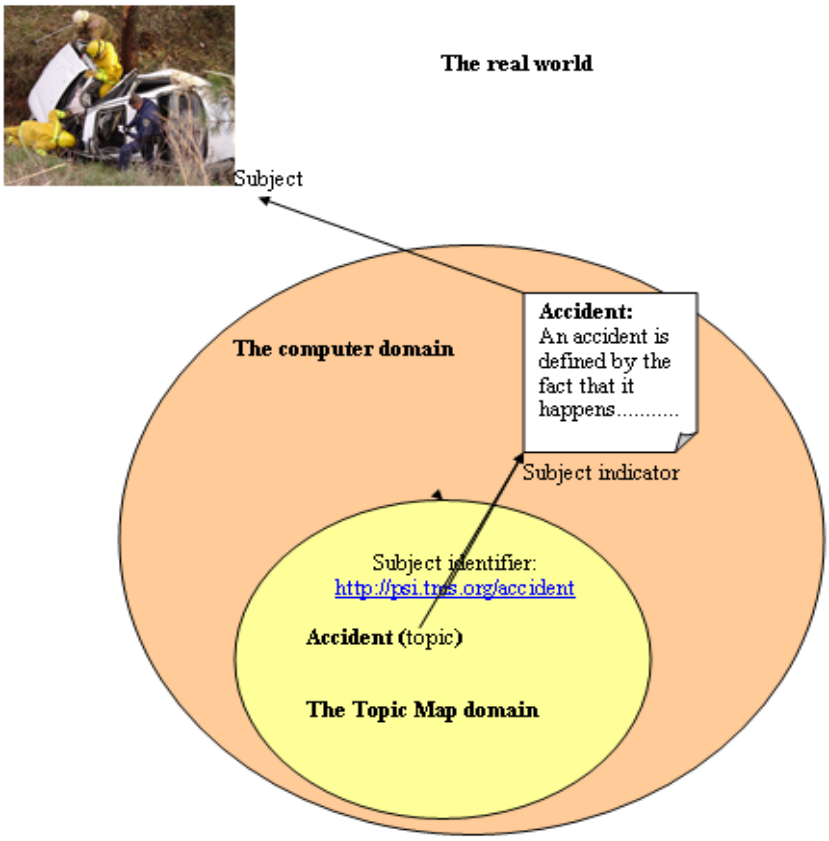


Figure 18 PSIs

The describing text indicates what is meant by the subject *accident*. Do we mean an accident where people are involved and maybe injured, or do we mean something unfortunate and more trivial that has happened. This is defined in the subject indicator. The next example demonstrates the same example in XTM syntax. The subject of matter is not addressable and therefore must be identified by a subject identifier.

```
<topic id="accident">
   <subjectIdentity>
     <!—the PSI - ->
      <subjectIndicatorRef xlink:href="http://www.psi.org/acc.xtm#accident"/>
    </subjectIdentity>
  </topic>
```

general PSIs, this way every topic mapper can make use of a common set. The OASIS Published Subjects Technical Committee Recommendation[37] is a co-operation between the OASIS consortium and the topic map community. They have developed a technical recommendation for published subject identifiers. The goal is to enhance the interoperability between topic maps by developing sets of PSIs.

For now, different systems can use different PSIs. To enhance interoperability across organizations, they can interchange the PSI repositories and make use of both. This way it is more likely that they can merge topic maps correctly when they agree on the topic identities.

There are 11 mandatory published subject identifiers that should be added to every topic map implementation. These ensure interoperability through a consistent behavior. These are; subclass, superclass, topic, association, sort, instance, class-instance, superclass-subclass, type-instance, type and occurrence [29]. There are 26 core subject identifiers in total [38].

Reification

The concept of reifying topics does not have a central position in this thesis and it is not essential for understanding the contents. The concept will be explained in detail in *2.3.5.4*. If a topic represents another topic this is called reification. This has a parallel to the process of representing subjects by using topics. The *Reifiable* class and the relationship to the *Topic* class is omitted in Figure 12 for simplicity. The topic has, in addition to the properties mentioned above, a [reified] property. This contains the topic map item that the topic has reified. The reified item will have a [reifier] property which refers to the topic that represents it. A reified item could be e.g. an occurrence or a topic name which are reified as topics.

Complete example

Below follows a complete example with all the concepts mentioned in this section. The *espa_train_accident* topic is an instance of the *accident* topic, which should be a topic in the same topic map to be correct.

```
<topic id="espa_train_accident">
   <instanceOf>
       <topicRef xlink: href="#accident"/>
   </instanceOf>
   <subjectIdentity>
      <subjectIndicatorRef xlink:href="http://www.psi.org/acc.xtm#espa_accident"/>
   </subjectIdentity>
   <baseName>
        <baseNameString>The Espa train accident<baseNameString>
        <! -- Form for sorting (sort name) -->
        <variant>
                <scope>
                        <topicRef xlink: href="#sort"/>
                </scope>
                <variantName>
                        <resourceData>Espa, train accident</resourceData>
                </variantName>
        </variant>
   </baseName>
</topic>
```

2.3.3.2 The Association

The next letter in the *TAO of Topic Maps* describes the *association*. The association expresses the relationship between one or more topics. This is what connects the topics in the topic map together and adds meaningful relations. The topics and associations constitute a semantic network or a knowledge map [25, 26]. The data model below illustrates that an association can have three properties, [type], [scope] and [association roles]. The association can have only one type, but many scopes and association roles. As Figure 19 demonstrates, the association role is what creates the connection between the association and the topic.

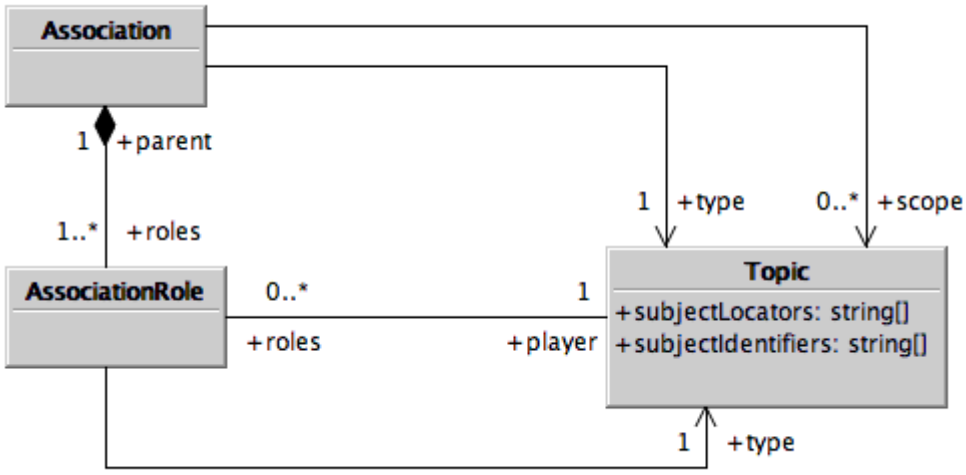


Figure 19 Association model

Adding context

Figure 20 can either be read from the left; Train456 was the place of an Accident, or from the right; Accident took-place-in Train456. By using roles, the direction is implicit communicated. *Train* plays the role *vehicle* and the role *accident* is played by *Espa-accident*.

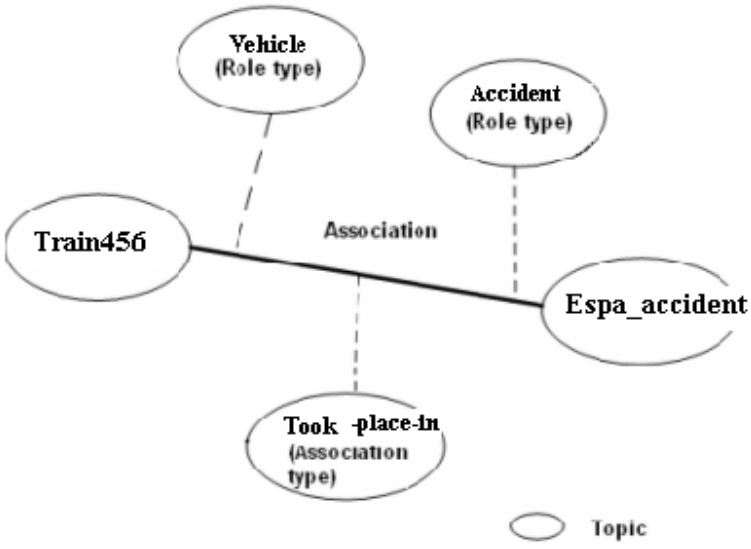


Figure 20 Association excerpt

The directionless association

A feature that makes topic maps flexible is the n-ary relationships.Associations can be unary, binary, ternary and n-nary. The binary is the most common and is also recommended to avoid too complex constructs [25, 26]. Instead of using directions, the association roles are utilized instead. An association roles consists of two topics, one is used as role and the other the player of the role. A syntax example of Figure 20 follows below. The association is typed with the *takes_place_in* topic.

```
<association id="location">
    <instanceOf>
      <topicRef xlink: href="#takes_place_in"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink: href="#accident"/>
      </roleSpec>
      <topicRef xlink: href="#espa_accident"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink: href="#vehicle"/>
      </roleSpec>
      <topicRef xlink: href="#train456"/>
    </member>
</association>

<!—topic types -- >
<topic id="vehicle">... </topic>
<topic id="accident">...</topic>
```

Reifying the association

An association can be reified by another topic. When an association is represented by another topic, it can be assigned characteristics that an association cannot have. Characteristics can be e.g. topic names and occurrences. To know which topic has reified the association the association has a [reifier] property that contains a reference to the topic.

2.3.3.3 The Occurrence

The last letter in the *TAO* is the letter *O* and stands for occurrences. An occurrence is the representation or the address of an information resource. In the policeman example from earlier, the topic *policeman* can be connected to a given article by having an occurrence attached.A topic links the information resource by using an occurrence. As associations connect topics together in the knowledge layer, occurrences link the resources in the information layer to the topics in the knowledge layer (*see Figure 4*). An information resource can be a document, image, web page, contents in databases and so on. Figure 21 illustrates that an occurrence can have [type], [scope] and [datatype]

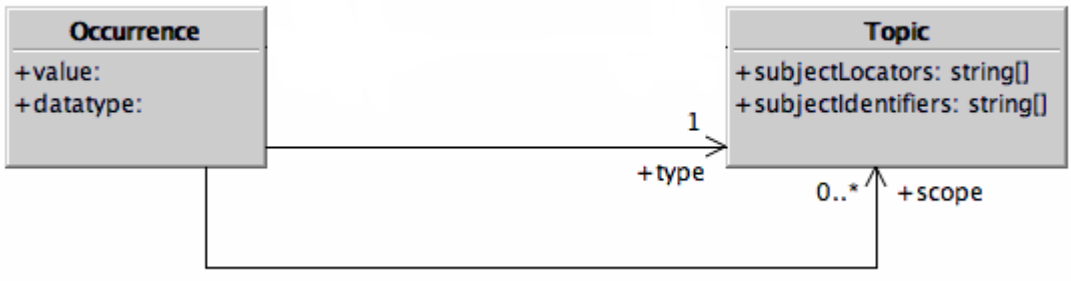


Figure 21 Occurrence model

External and internal occurrences

The resources are not always located outside the topic map. The occurrence can also be located inside the topic map. This is called an internal occurrence.

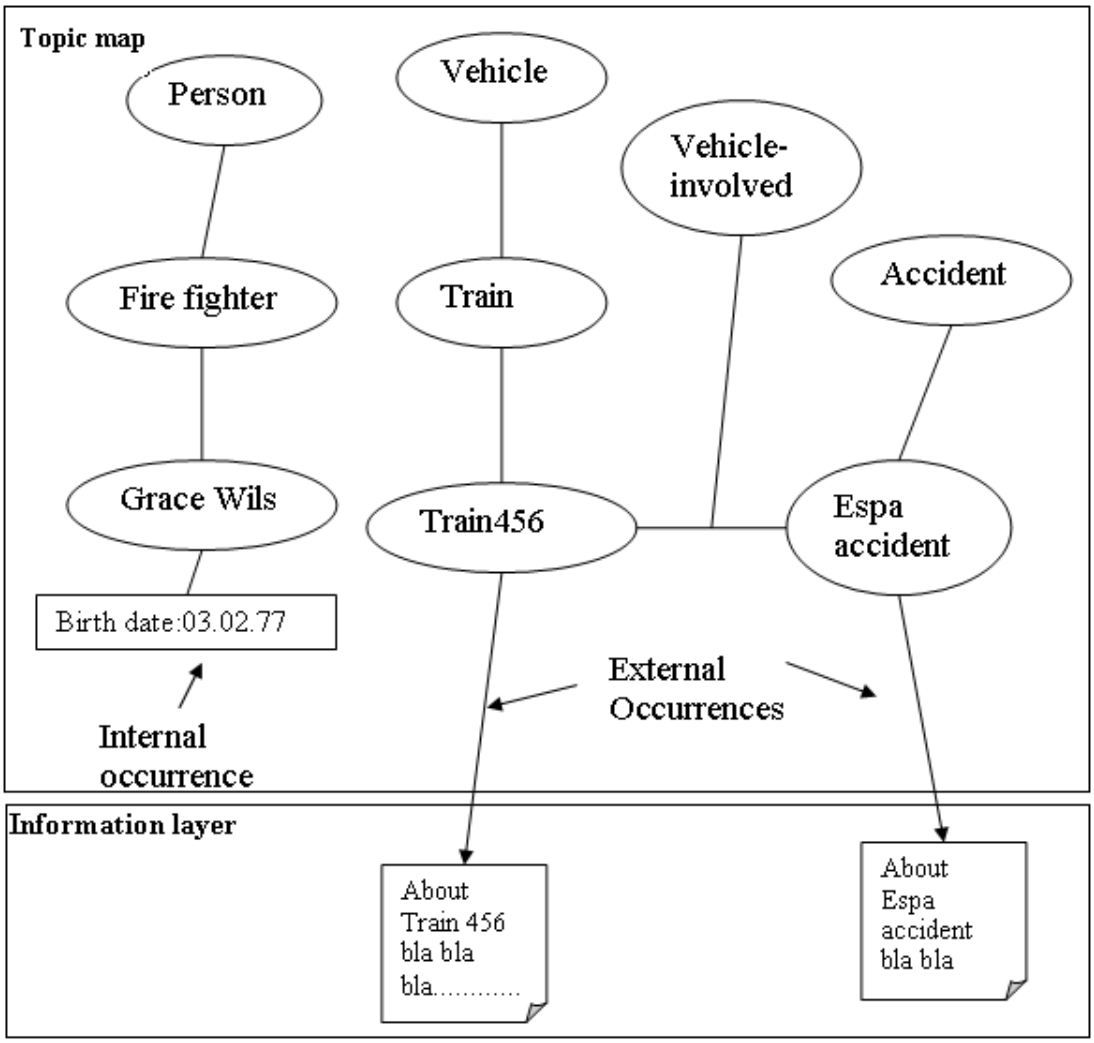


Figure 22 Occurrences

The topic map expresses an internal occurrence by using inline character data. This data represents extra information attached to the topic. An example of this are dates; this is not a characteristic of a topic that can be linked to an information resource. Instead, this can be attached directly in the occurrence description tag. The occurrence is either an URI, indicating the location of the resource that should be connected to the topic of matter. Alternatively, the occurrence *is* plain text that says something about the topic. Figure 22 demonstrates the two types. An occurrence type that references an external resource can be *accident_report*, *web_page*, *paper* or *draft* and so on. An internal occurrence could be *date_of_accident*, *date_of_birth* and so on. Another possibility is to use associations to connect the topic and the occurrence together. This could be associations like, *discussed-in* or *mentioned-in*. The example below illustrates a topic with both internal and external occurrences [38].

```
<topic id="espa_train_accident">
    <occurrence>
       <!—the type of occurrence -->
       <instanceOf>
            <topicRef xlink: href="#date-of-accident"/>
       </instanceOf>
       <! — internal -->
       <resourceData>01-05-2001</resourceData>
   </occurrence>
   <occurrence>
      <instanceOf>
           <topicRef xlink: href="#accident_report"/>
      </instanceOf>
     <! — external occurrence -->
     <resourceRef xlink: href="http://www.emergency./reports/espa.pdf"/>
   </occurrence>
</topic>
```

2.3.3.4 Advanced Concepts

After this basic introduction to the TAO of Topic Maps, some remaining concepts deserve an introduction. The process of merging topic maps is discussed more thoroughly in Chapter 5.

Reification

As mentioned earlier in the brooding sections, it is possible to reify a small part of the topic map, in other words; create a topic that represents a topic map construct. Topic map items that can be reified are;

- topic names
- variants
- associations
- association roles and
- occurrences

By using the reification mechanism, the new topic can be assigned characteristics. In addition, the topic map will consist of several knowledge layers within the topic map. It will enable multiple levels of knowledge representation within the topic map itself [29]. The act of reification does not imply that the original topic map constructs are deleted. One example is to reify an association to *one* topic.

The reified:

```
<association id="espa-accident_in_train456">
    <instanceOf>
      <topicRef xlink: href="#takes_place_in"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink: href="#accident"/>
      </roleSpec>
      <topicRef xlink: href="#espa_accident"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink: href="#vehicle"/>
      </roleSpec>
      <topicRef xlink: href="#train456"/>
    </member>
</association>
```

The XTM example below demonstrates a reification of the association *espa-accident_in_train456*. The original association cannot be given names or occurrences. Now, the reified topic can be assigned characteristics like *names* and *occurrences*.

The reifier:

```
<topic id="accident-takes_place_in-train-topic">
    <subjectIdentity>
        <!—reifies the association -->
        <topicRef xlink: href="#espa-accident_in_train456"/>
    </subjectIdentity>
    <!—assigned name -->
    <baseName>
        <baseNameString>Espa accident in train456</baseNameString>
    </baseName>
```

```
    <! -- Occurrences may go here -->
  </topic>
```

It is not possible to let a topic reify another topic. If doing so this would indicate that they are in fact describing the same subject and hence should be merged [38]. It is possible to reify the topic map itself, to assign characteristics to it.

Scoping Scope is about adding context or perspective to a topic. To do this one can create a topic used as scope and add the scope element to the desired item.

> *"A Scope is set of Topics that defines the extent of the validity of the assignment of a Topic Characteristic to a Topic[29]"*

The scope element can be assigned to, topic names, variants occurrences associations and associations roles. An item without a scope has an unlimited validity. This is known as the *unconstrained scope*. The purpose of scope is to avoid ambiguity, to control the vocabulary or to be able to add several names to a topic. A scope can be utilized for ranking, filtering or selecting information of a topic in a certain context/perspective. When a topic has several topic names, it can be very useful to filter out the unwanted ones. Filtering is defined as,

> *the removal of irrelevant topic characteristics* [39].

The next example demonstrates the use of scope in the topic *accident*. The topics, *en* and *no* are topics that are used for scoping purposes.

```
<topic id="accident">
    <! - base name for English -->
    <baseName>
        <scope>
            <topicRef xlink: href="#en"/>
        </scope>
        <baseNameString>Accident</baseNameString>
    </baseName>
    <! -- baseName for Norwegian -->
    <baseName>
       <scope>
          <topicRef xlink: href="#no"/>
       </scope>
      <baseNameString>Ulykke</baseNameString>
    </baseName>
</topic>

<topic id ="en">
    <baseName>
        <baseNameString>English</baseNameString>
    </baseName>
</topic>

<topic id ="no">
   <baseName>
        <baseNameString>Norwegian</baseNameString>
    </baseName>
</topic>
```

Creating topics used for scope makes the topic map more reusable. Several topics can share a scope. This avoids redundant information items and promotes low coupling. The scope element is also a good tool for avoiding ambiguity between topics with equal topic name. The

topic *Paris* could mean the capitol of France, a character in Romeo and Juliet or a city in Texas. When adding scopes like *capitol*, *character* or *city* to the topics, the correct context adds the correct meaning without the use of subject identity. In addition this prevents the topics from merging [29].

Ranking is the ordering of characteristics according to their relevance. This could be ordering the variant names of a person in the scope order; *last name*, *first name*, *sorted name* and *call name*. Selecting, defined as choosing the most relevant characteristic in a given situation. This could be selecting the sorted name or the full name of a topic.

2.3.4 Topic Map Engines

When developing topic maps software, the most important tool is the "engine". Software that processes and optionally persist instances of the topic map data model is so-called "engines". There are some existing stand-alone engines as well as frameworks. They exist in two categories; free, open source and proprietary/commercial ones. This section gives a short overview of the existing software, but first the Topic Map API needs an introduction.

2.3.4.1 TMAPI

There is a de facto standard API, called the Common Topic Map Application Interchange Interface [40]. This API consists of mostly interfaces, which serves as a common interface for different topic map implementations. This ensures the developers of different applications and engines a common "language". The implementations can then easily be connected and be extended. In addition this ensures a consistency in the topic map development community. The class diagram in Figure 23 illustrates the hierarchy. All of the classes are interfaces. When starting using the API an instance of the *TopicMapSystemFactory* class must be instantiated. This class locates the implementation of the interfaces. A TopicMapSystem can contain several topic maps.

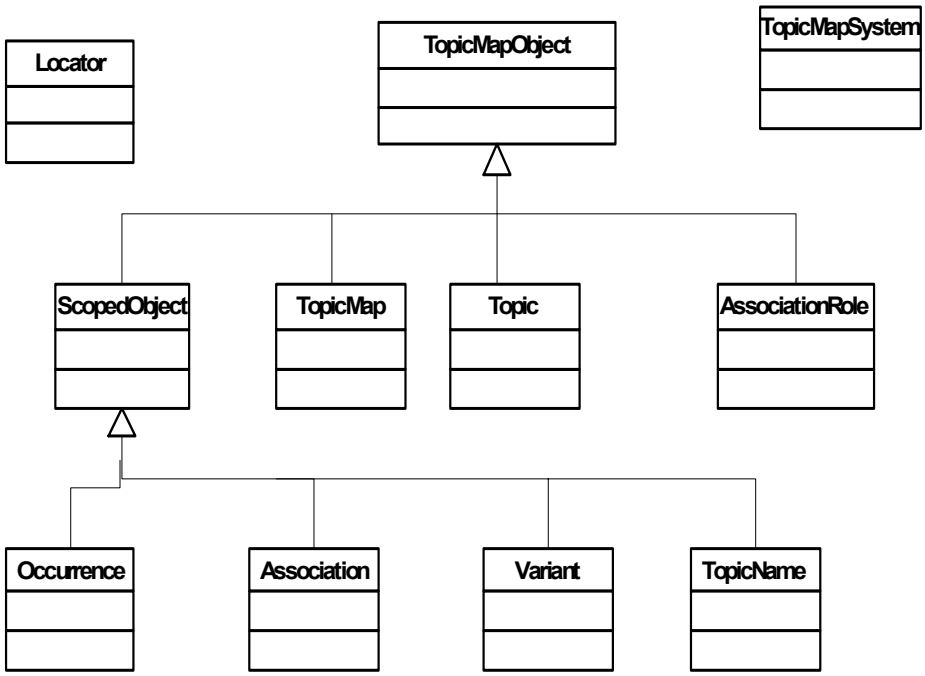


Figure 23 TMAPI class diagram

The API has also interfaces for indexes and exceptions. The API is written in Java, however there are projects developing API in other languages. AN open-source project called TMAPI/PHP has developed an API for PHP programmers based on the TMAPI. In addition a CPAN module in Perl has also been released. The module enables reading and writing of XTM files and AsTMa resources, and manipulating and querying topic maps via an API.

2.3.4.2 Tools

There are several pending projects and different languages/syntax under development. This section lists the most important ones.

TinyTIM
TinyTIM [41] is a light-weight in-memory free Topic Map engine. It is developed by Stefan Lischke and can be downloaded at SourceForge.net's project pages. This implementation uses the TMAPI and a small JAR footprint.

XTM4XMLDB
This open source topic map engine uses native XML databases such as eXist or Xindice as the persistent store. It implements the TMAPI.

ZTM
This open-source engine is written in Python and aims at enabling distributed development and maintenance of topic map driven semantic webs.

The Omnigator
As a shortcut and as an introduction it is possible to download Ontopia's own showcase; the Omnigator [42]. Ontopia created the Omnigator for new users to become familiar with the Topic Map standard. The user can use this tool to debug and test-drive a topic map. The Omnigator is omnivorous and can eat anything that can be viewed as a topic map (this includes RDF). The tool supports every aspect of the Topic Map standard [43]. This engine is a part of the Ontopia Knowledge Suite.

TM4J
The Topic Map 4 Java engine [44] is a part of the TM4J project which is also located at sourceForge.org. This is the leading topic map engine in the community. The engine is written entirely in Java and is based on the TMAPI. The engine supports TOLOG and the import of XTM and LTM syntaxes. In addition, it supports persistence of topic maps in different databases. The project aims at developing robust open-source tools for processing topic maps. Other tools they develop are; Panckoucke, TMNav and TM4Web [45].

Perl XTM
Perl XTM [46] is an engine that can read/write XTM files, read/write AsTMa resources and manipulate and query topic maps via an API.

TMCore05.net
This is a commercial and proprietary Topic Map engine. It is designed for the .NET platform as the name indicates [47].

OKS
OKS is an enterprise knowledge suite developed and owned by Ontopia. Is contains an API for building any kind of topic map application. This enables building web-delivery

applications e.g. portals and intranets, editing applications, and it can easily convert data to topic maps.

The SDK (Software Development Kit) implements the full topic map standard (Original ISO 13250 and XML Topic map Maps 1.0). It supports three interchange syntaxes; HyTM, XTM and LTM. This API supports all form of handling topic maps. This includes reading, writing, creating, modifying, and traversing. In addition to this the SDK supports high-level utility functions like; *indexing services, merging services and filtering and selection services*. There are two possible implementations, either in-memory or RDBMS storage.

The engine implements also the Topic Map Constraint Language (TMCL). For now it supports the Ontopia Schema Language (OSL), Ontopia is committed to support the ISO TMCL when it is finished. Having a schema language makes it possible to validate the topic map semantically; it simplifies user interfaces (schema-driven editing) and enables optimizations. There is also a query engine for performing queries directly on the topic map structure. The Topic Map Query language implementation uses TOLOG as query language. When ISO Topic Map Query Language (TMQL) is finished, Ontopia will make sure the engine supports this standard. This simplifies application development and improves performance [30].

2.3.5 Application range

After introducing and elaborating the concepts and mechanisms that Topic Maps can offer, some application examples are in place. The list below indicates some example applications,

- Concept-based e-learning. This is considered used in schools for learning purposed. The well-known brain-map is close related to topic maps. By visualizing the contents of a course in a topic map, the students can easier grasp the relations and essence.
- Organization large bodies of information in portals and web sites.
    - http://forskning.no (Norwegian Research Council)
    - http://forbrukerportalen.no/ (Norwegian Consumer Association)
    - http://matportalen.no/ (Dept. of Agriculture food portal)
    - http://itu.no (Dept. of Education portal about IT in education)
    - http://www.hoyre.no (Norwegian Conservative Party)
- Managing distributed knowledge and information.
- Capturing organizational information

The University of Oslo have experimented with the use of Topic Maps. the library has tested the usage of Topic Maps in BIBSYS[48]. The tests showed that the relations and the topic map structure gave the user several possible navigation routes. By describing the relations, the user received help in choosing the right direction. They concluded in the report that the technology was very suitable for retrieving information, and hence was a solution to the findability issue. Another project called *Houston Emnekart* [49] has implemented a system for keeping track of the operating computers (USIT) at the University of Oslo. The topic map is used to display the relations between the different services and to document the services.

It was easy to locate web-based Topic Maps applications, in addition to stand-alone applications [50]. I did not manage to find examples of distributed Topic Maps systems that used merging. It seems as if the field of Topic Maps has focused, primarily on Topic Maps as a navigation and knowledge representation layer. The usage is typically within an enterprise or an organization and aims at structuring organizational memory and assets. The vision of "Seamless knowledge" and the heart of Topic Maps, merging, seem to be in the shadows for the moment.

2.4 Topic Maps vs. RDF

One of the requirements in the Ad-Hoc InfoWare project was using ontologies as a means to share information across domains. The RDF [3] and Topic Maps are both knowledge representation technologies can both represent ontologies and make use of shared vocabularies. Since they both could be applied as knowledge representation standards according to the requirements in the Ad-Hoc InfoWare project, this chapter will shed some light on some of the differences between the standards. I will also try to argue the choice of Topic Maps.

2.4.1 RDF

RDF (Resource Description Framework) is a knowledge management framework for representing information on the Web. It has an abstract XML-based syntax for expressing the concepts of the framework. It is a graph-based theory and uses binary relations between the information items. A first draft of the RDF specification came in October 1997. It started as a standard on top of XML, for encoding metadata (data about data). The purpose was to connect metadata to Internet resources in RDF expressions, called *triplets* [51]. Figure 24 demonstrates a simple triplet.

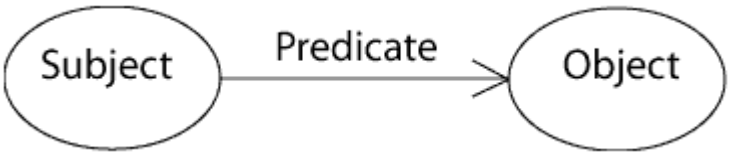


Figure 24 RDF triplet

RDF is based on a logical and mathematical graph theory. A set of triplets constitute a graph. A triplet consists of three different concepts, hence the name. These concepts are *subject*, *object* and a *predicate,* also expressed as *resource, value* and *property*. The property concept is an indicator of the relationship between the subject and object. This relationship has always the same direction. The use of URI for identifying the vocabulary makes the framework reusable.

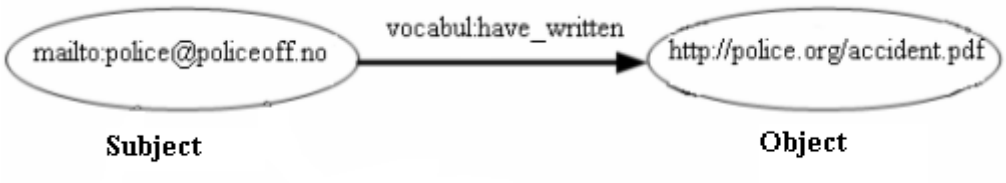


Figure 25 RDF statement

As Figure 25 demonstrates, the *subject* is defined by the URI *mailto:police@policeoff.no*. The *objects* are referenced with the URI; *http://police.org.accident.pdf* .The subject has one predicate; *have written*. RDF can express more by using the DAML+OIL or the Ontology Web Language (OWL), which is an extension of RDF. This may impose further semantic conditions, like; MUST, MUST NOT, SHOULD and MAY to the RDF statements [14]. OWL is a language used for publishing and sharing ontologies.

In 2004 the scope of RDF involved into something greater, from simply attaching metadata to web resources it started expressing things in real life, classes, people etc more similar to an ontology model [51]. RDF has now become a universal format for describing and interchanging data on the Web. In February 2004 the World Wide Web Consortium (W3C) released the RDF and the OWL as W3C recommendations in the Semantic Web initiative [17].

The RDF specifications provide today a lightweight ontology system to support the exchange of knowledge on the Web. The standard supports co handling between applications, by making the information machine-understandable and processable. RDF can be used in many areas; e.g. in resource discovery to improve search engine capabilities, in cataloguing to describe content, digital library, intelligent software agents, in content rating and many more. RDF is an important part of the Semantic Web [3]. The Semantic Web initiative started out in 2001, and the RDF was a part of this. The Semantic Web is an extension of the existing Web, and gives information a well defined meaning, which makes possible cooperation between computers and humans [17]. This project is currently under the supervision of Tim Berners-Lee. The idea is to use RDF to describe web pages or documents.

2.4.2 A comparison

This section is based on [52] where nothing else is given. The two technical communities worked concurrently on the two standards in the late 90s. They did not become aware of each other until early 2000. At that time both standards was in for approval at ISO (Topic Maps) and W3C (RDF). According to Lars Marius Garshol, they could have merged if they had been aware of each other at that time. Today RDF and Topic Maps find themselves in different communities and with a different family of standards. It will probably be both politically and technical impossible to merge the two.

The family of standards
RDF and Topic Maps have different set of standards. Figure 26 illustrates the mapping between them.

Figure 26 Families of standards

The two standards have two different goals, which explain many of the different solutions and approaches in their specifications. RDF is intended for attaching metadata to web resources and eventually contributes to the semantic web vision. Topic Maps was created to support high-level indexing of information items [9].

	Topic Maps	RDF
Merging	Yes	No
Verbose	No	Yes
Scoping	Yes	No
Relations	n-ary, no directions	Binary, with directions
Flexible	Yes	No
Complexity	Yes	Less complex
Reusability	Yes	Yes
Theory	--	Graph-based
Syntax	XML based	XML based
Supports thesauri	Yes	Yes, by using OWL
Handle non-addressable resources?	Yes	No
A large community for support?	No	Yes, bigger
Based on	Identity-based technology	Identity-based technology
Perspective	Human understandable	Machine perspective
Vision	Seamless knowledge	Semantic Web
Information processing	Querying, navigating	Reasoning
Applications	Some applications	Yes many applications

Table 1 Topic Maps vs. RDF

According to Table 1, the standards have different strengths and weaknesses in addition to many similar concepts and ideas. They both try to add semantics to information items and use shared vocabularies to promote reusability. They represent information through the means of a XML-based syntax. The different syntaxes have some resemblances; subjects/topics are connected to objects/topics through predicates/associations. However, the details and applications are quite different.

One of the reasons for choosing Topic Maps is the fact that it is a knowledge representation standard aimed at connecting knowledge together, especially across different systems and organizations. RDF is originally a framework for representing resources on the web. Moreover, the goal is to add semantics to *every* web page on the Web. RDF has a more

limited model for expressing the relationships between information items, while Topic Maps have many more complex and additional functionalities like, scoping, typing and n-ary associations. By the use of OWL, RDF can express a much more detailed ontology. However, the focus on merging Topic Maps in distributed systems gives the Topic Maps standard an advantage in the competition between the two. Topic Maps objective is to achieve seamless knowledge between the systems or organizations that use it. This is one of the most important requirements in the Ad-Hoc InfoWare project.

Mapping RDF and Topic Maps

RTM (RDF to Topic Maps mapping)[53] is a vocabulary that can be used to describe the mapping of an RDF vocabulary to topic maps in such a way that RDF data using that vocabulary can be converted automatically to topic maps. This allows RDF vocabularies to be used both as RDF vocabularies and as topic map vocabularies. This effort can build a bridge between the two competing standards and instead merge the strengths and potential of the two together.

3 Related Work

There are few applications of topic maps compared to other knowledge representation technologies like for instance RDF. Topic Maps has been an international standard since December 1999 [4, 54]. A newer version came in 2003. It has taken time to develop the different languages for retrieval, querying and supporting the Topic Maps standard. Many standards and languages are under construction and this keeps the developers waiting. In addition, the reputation and rumour has moved somehow slow. It is not until recently that courses at the Department of Informatics have started lecturing about Topic Maps as a new paradigm in knowledge representation. As a comparison, Google.com gives 77 million results when asking for "RDF" and only 1 million hits when asking for "Topic Maps". However, the distribution and the applications are growing.

When it comes to the usage of Topic Maps on mobile devices there are two projects. The first one is the Shark project which was the origin of the Mobile Topic Viewer (MTV) [5]. A second example is a project group at the University of Mannheim [55] who has proposed system architecture for creating query tools on mobile devices that implements the Topic Maps standard. Another relevant work is the SIM project which deals with the problem of subject identification when a shared vocabulary is absent [56]. A shared vocabulary and the use of PSIs is a presumption for merging correctly. Below follows a description of some of the interesting research projects and existing applications that have relevance to the thesis.

SHARK

Shark is an acronym for "(Mobile) Shared Knowledge". The Technische Universität (TU) Berlin in Germany established the project in 2001 by the initiative from Thomas Schwotzer. The idea is to create a technology platform that enables the distribution of knowledge with the use of mobile devices. To do this the Shark project uses knowledge bases (knowledgebase) with Topic Maps as format and SyncML for synchronizations between knowledge nodes. The special aspect of the project is the usage of mobile extensions of knowledge bases. A knowledge base is a special kind of database for knowledge management [57]. The platform will consist of a Topic Map application based on the Ontopia Knowledge Suite that runs on the server-side. The mobile devices use a mobile implementation of the Topic Map API, with a *RecordStore* backend for Java ME [58]. This is used to parse and handle the topic maps in a mobile shared knowledge environment [59].

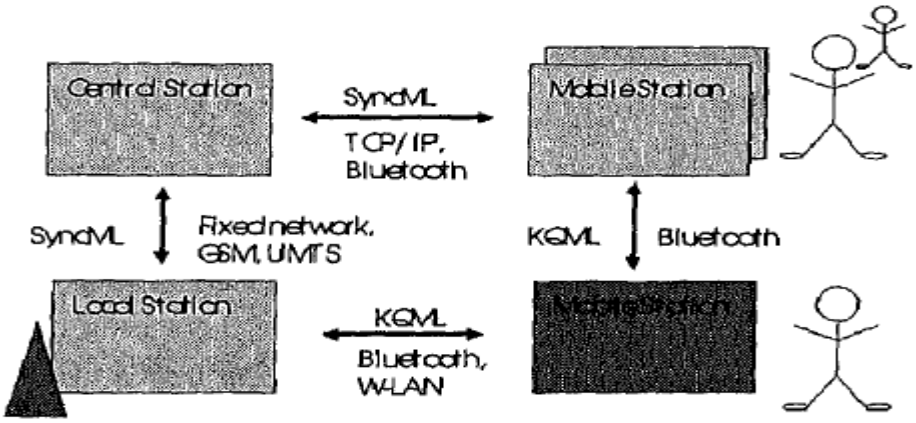


Figure 27 Shark communication model

Figure 27 demonstrates the communication in the Shark model. The platform will support organization, synchronization and exchange of knowledge within a mobile user group, or between different user groups. The system wants to extract knowledge from a knowledge base and insert it into another. To do this they introduce a new concept called *knowledge ports* [5].

The project resulted into a PhD. and a prototype Topic Map engine for mobile devices. The engine does not implement merging, but rather uses synchronization. Marcus Walla continued the work on the mobile engine and the MTV was born. This engine is the core of the technical implementation in Chapter 6.

Thomas Schwotzer has become head of research at Neofonie and further development is planned. The mobile topic map engine should be launched as TM4J2ME. The vision is to implement a demo version of a mobile P2P network based on the MTV and the Bluetooth technology.

TMShare
The TMShare [60] is a peer-to peer information-sharing application based on Topic Map technology. The TMShare application environment consists of a group of interacting peers. Each peer in the group provides the other peers with access to the information it manages by presenting subsets of the information as topic map fragments in response to requests. The topic map fragments received from remote peers are cached locally and presented as if they were merged with the locally held topic map information. Cached results can then be provided to other peers in response to their requests.

Topic Map query tool based on J2ME
At the department of Informatics at the University of Mannheim, system architecture has been proposed for a topic map query tool on mobile devices [55]. They stress the importance of focusing on mobile technology for supporting knowledge management. They have developed prototypes for Java-based mobile tools that exchange XML with a Topic Map server. The TM4J has been used as engine on the server. The project focuses on the tremendous boost in the development of new mobile technology the last years. Functionality, capacity and availability have increased considerably.

The SIM project
The Subject Identity Method (SIM) [56] project tries to solve the subject identity problem in a distributed system when a shared vocabulary (ontology) is absent and there are no PSIs. SIM involves a complex algorithm that uses the structure in the topic map to establish the equality of two distributed topics. When merging topic maps from different system, a problem occurs when they do not have a common set of PSIs. The SIM method should decide the equality between the topics instead of PSIs. When a common set of PSIs are absent, the SIM method could be a solution. This is still ongoing research and there are unsolved issues that should be dealt with. If the emergency scenario should be extended to co-operation across national boundaries, this could be relevant.

TinyTIM
TinyTIM is a lightweight free TMAPI implementation. It is very suitable for learning purposes. Since the merging functionality was implemented his engine worked as a study case before the technical implementation in this thesis. It turned out to be instructive and a good starting point before the implementation. After studying for a while, some obvious shortcomings appeared in the merging algorithm. They are listed below,

- No multiple subject locators equality test

- Do not consider variant names
- Adds association roles only if the merging occurs within *one* topic map.
- No checking for duplicates in associations, simply adds the associations.
- No scope checking in mergeByTopicName feature
- Correct references are lost when adding types to a topic, topic A is assigned topic B2 from topic map tmB as type. A topic should not refer to a topic in another topic map.
- Checks equality by comparing objects in different topic maps, these objects will never be equal.

The details of the implementation are a matter of how big and complicated the implementation should be. The implementation uses the Java SE library and has a different data structure compared to the MTV. These reasons in addition to the fact that the engine is incorrect; there was no possibility to transfer the merging implementation from tinyTIM to the MTV.

4 Use case and Requirements

The Ad-Hoc InfoWare project has worked out some requirements for the knowledge management module. These are the basis of the requirements of this thesis. There is a set of prerequisites that must be present to ensure correct information exchange in the sketched out use case. In addition, there is some delimitation to the technical implementation. These are presented in this chapter.

4.1 Requirements

We have made a list of the prerequisites that must be present to accomplish a correct information exchange during a rescue operation. They have been slightly adjusted to fit the profile and characteristics of the chosen Topic Map technology.

- The personnel have in front, discussed and elaborated a plan for coordination and action on the scene. This includes setting up a Mobile Ad-hoc Network
- The emergency personnel must have the technology necessary. This involves Java compliant mobile devices, Topic Map engines and elaborated applications for the purpose.
- There must be a co-operation between the involved organizations in forehand to develop a common ontology, which expresses the emergency domain. This ontology will be the connection point between the organizations.
- Each organization must develop a topic map for expressing their domain. The common ontology must be a part of this topic map. The topic maps should be based on the XTM 2.0.
- A set of common PSIs, used by the different organizations must be elaborated. This can be accomplished by establishing a common repository. These PSIs must be used as subject identity in the topic maps. This enables a more precise merging.
- Sharing information across boundaries by using the merging process in topic maps
- It is set up conditions for easily merging between the units.
- Each rescue worker that possesses a mobile device can easily receive and send information in the form of topic maps.
- Each resource-weak device can store the part of the topic map that is crucial for the purpose, and add information needed by importing and merging the topic map of another domain.
- The ontology is read-only but the personnel can add information items to the topic map.
- To avoid sharing sensitive or private information scoping and/or typing can be applied.
- To avoid information overflow the topic map can be filtered by typing.
- To receive the information in the correct context and avoid ambiguity the information items can be scoped

These requirements will work as a staging area for the rest of the thesis. The prototyping will follow these guidelines and strive to be a realization of these items.

4.2 Scenario with Topic Maps

This suggested use case scenario will attempt to illustrate the utility value of utilizing Topic Maps. Figure 28 illustrates the co-operation between the different organizations on the scene. It is based on the assumption that merging on mobile devices is accessible.

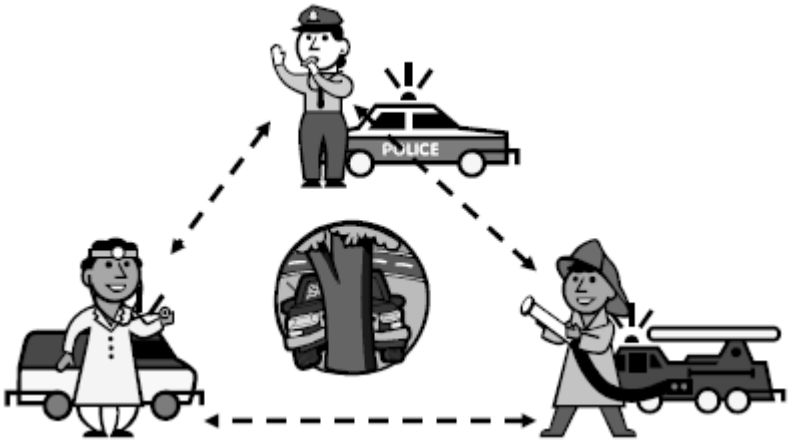


Figure 28 Co-operation across domains

The sequence of the supposedly solution scenario:

1. Elaborate a shared ontology (by using topic maps) for the emergency domain. Every organization involved uses this ontology. There are today some ontologies elaborated for e.g. genomic and anatomy. A suggested project for developing an ontology for the emergency scenario exists here.
2. Every organization has its own topic map based on the shared ontology. The different organizations have in front exchanged the necessary information, i.e. they have worked together on using the same set of PSIs for subject identity in their topic maps. This will assure that identical topics will merge and have characteristics from both topic maps.
3. On the emergency spot the coordinator in charge have a laptop available with all the potential information needed. This includes the topic maps. The other personnel have mobile phones, equipped with a topic map specific for the organization they work for
4. During the rescue operation, the e.g. ambulance personnel mark the injured persons and quickly insert the information on the mobile device. The worker can send a message to every node in the MANET. The other nodes in the network intercept the message and the workers will either be alerted by an alarm sound or will by itself download the latest changed topic map and perform merging.

Gather information

This involves rating the level of injury of each individual involved in the accident. The level-system will be a very easy and short scale; green is small or no injury, orange implies the damage is of some importance, but not critical and the last colour *red,* indicates a high level of injury which needs immediate attendance.

Mark the injured persons with e.g. censors after a short examination. E.g., the EO or the leader of order can perform this. The censor could be shaped as a bracelet and therefore be easily attached to the injured person. The bracelet can have its own unique id, and the mobile device (unit used) can intercept this. The censor will indicate by colour the level of injury.

The colour is set by using the handheld device. The colour ranges from green; meaning no or very small injury, orange indicates level of injury as medium and red means critical. When a rescue personnel attaches a bracelet to an injured, the id and the profile pops up on the screen. This simple interface enables the registration of data for the person. A simple colour choice, changes the colour of the bracelet. This way everybody can see the level of injury, not only the personnel carrying a mobile device. Registration and marking of the injured persons is only performed when there is a need. It will be practical at extensive rescue operations, which involves many injured persons.

Another use case is to insert the information about injuries in the topic map. The information can then be shared in the MANET.

4.3 Usage of Topic Maps Concepts

The Topic Map standard has many valuable functions and mechanisms that can be applied in the emergency scenario. I will go through the most essential concepts and demonstrate how Topic Maps can contribute.

4.3.1 Scoping

The topic maps loaded into the mobile devices will probably not be of very great size or complexity. Quick response and access to data will be of higher priority then large amounts of data. There will probably not be any time for complex queries or searching for information by scope. The data that is needed must be easy available and preferably be worked out in advance so that the user can have few but critical possibilities ready for use. According to the use case scenario, we see some interesting usages of the scope concept. One of the most important aspects is security. The list below lists some ways an information item can be scoped in the emergency scenario.

- **Access level:** e.g., *security* (unclassified, classified.), *user level* (beginner, intermediate, expert)
- **domain:** e.g., *subject area* (police-domain, medicine, emergency), *historical period* (ancient, classical, medieval)
- **natural language:** e.g., English, French, Norwegian
- **resource location:** e.g., offline/online
- **viewpoint:** i.e., which authority asserts the relevance of the resource to the topic

Information overflow
Another issue to take in consideration is the aspect of information overflow. The whole concept of using the topic map standard is to avoid this phenomenon. When merging other topic maps this could lead to too much information that is not in the interest of the user. So, to avoid this there could be made possible to only send over parts of the topic map, when requested from another user. This will elaborated in advance, which parts of which topic map could be interesting to the other organizations.

Security
When it comes to sharing data across organizational boundaries, a question of sensitive data arises. Each organization must have the possibility to conceal information. This is where the scope function comes in. This can be used to filter information, as mentioned earlier. The sensitive data can be scoped *private*. When querying the topic map each organization could have its own code for getting the *private* information items in the topics. When a user has imported another topic map and they are merged, she will not be able to query for certain characteristics in the topics. Private occurrences can have a scope of unknown name/id. A solution is to scope the desired names or occurrences with *public* or *private* to indicate the access level. Another solution is to import topics that have *public* scopes from other topic maps. This way the user does not have to import topics with no access. The application developer in cooperation with the users will solve these questions.

Another usage is to describe whether an occurrence is offline/online. This could be very valuable in a MANET, where the network is not 100% reliable. In addition, scoping could define the domain the knowledge is a part of. If some topics were used in both the *fire fighters domain* and the *medicine* domain, it would be of high importance to separate the topics by adding scope.

4.3.2 Typing

Since a topic cannot be scoped, there is a need for classifying on topic level. Typing is a mechanism in Topic Maps that allows a topic to belong to a certain class. The class can be anything, e.g. *person*, *injury-type, flammable-fluid* and so on. Scope marks the characteristics of the topic, while typing can mark the topic itself. When a topic should be concealed to other organizations it can be typed with *public/private*. When querying for topics the typing function has some of the same advantages as the *scope*. Where scope makes it possible to query for an information item in a certain context, typing can embrace many topics that belong to the same class. When querying the topic map, the user can ask for every topic of type *injured-person*. Then, she will receive information of every injured person of a given accident. She could make the query more detailed and ask for every injured, female person with burn-damages. These queries should be elaborated in forehand. The user will not have time to create new ones on a hectic emergency scenario.

4.3.3 Merging

To benefit from the act of merging topic maps, a common upper ontology is needed. This could be a topic map describing the super classes of an emergency scenario. These topics will be a part of a common ontology and connect the different domain specific topic maps together. Topics like; *accident*, *train_accident*, *location*, *electronic document*, *injured person*, *person*, *vehicle* and so on.

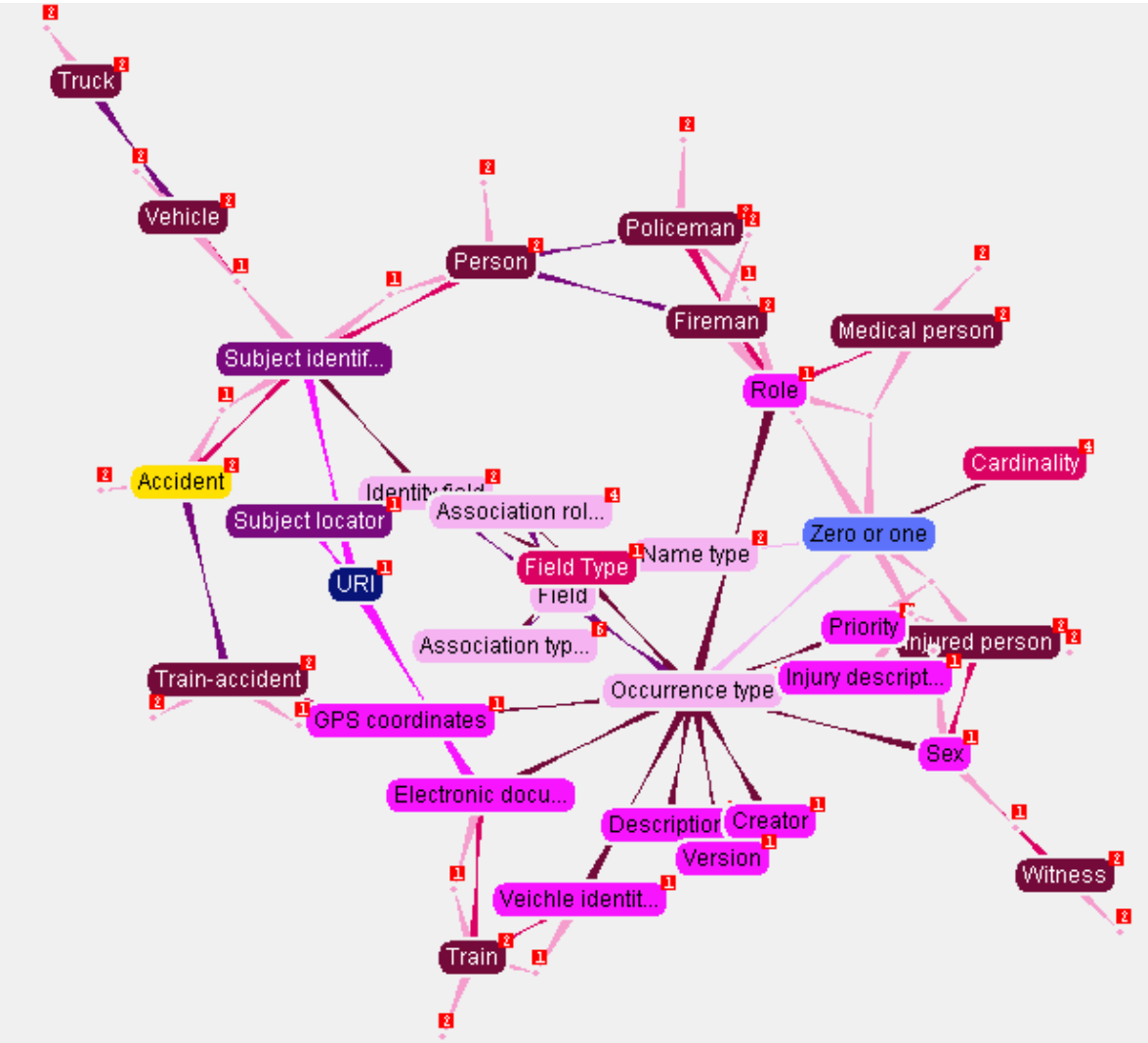


Figure 29 Ontology excerpt

Figure 29 illustrates an excerpt from one of the topic maps, elaborated by Samuel Vigdal and me. The picture displays the ontology in the topic map. The topic map is displayed in Vizigator [61], a visual topic map tool.

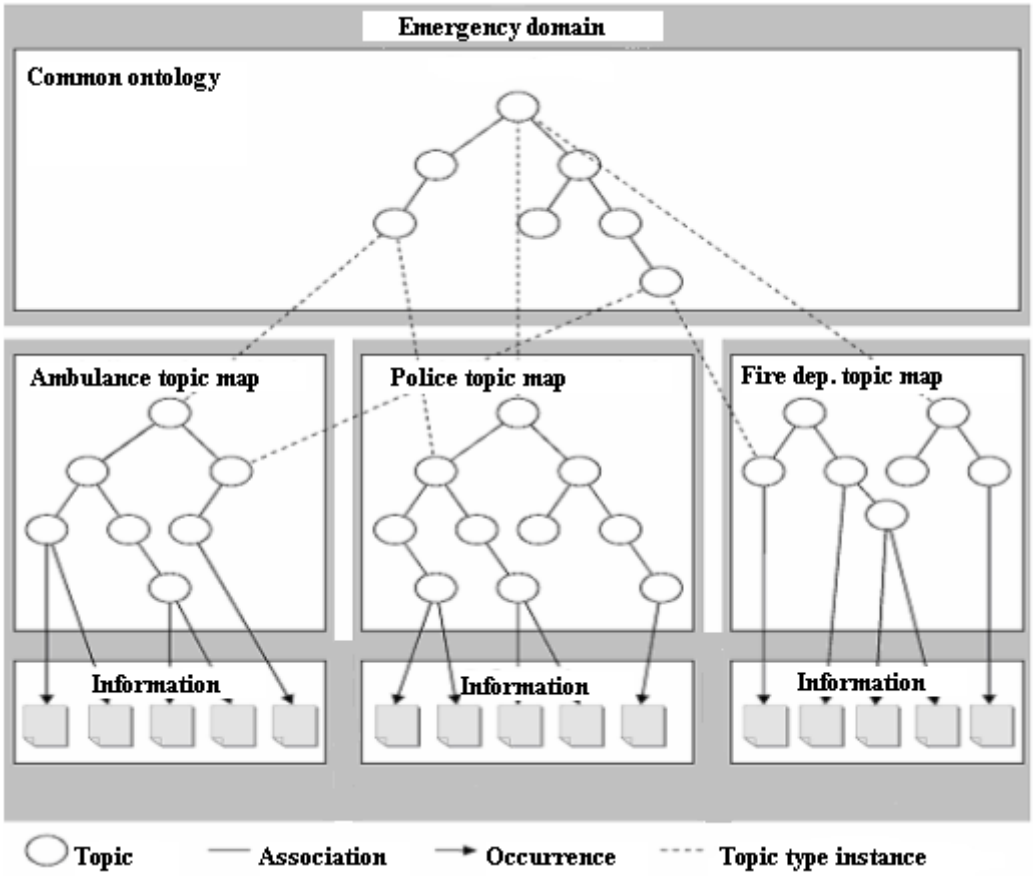


Figure 30 Emergency use cases

Figure 30 illustrates the application of a shared (common) ontology. The ontology will contain general emergency-specific topics that are useful for all the organizations involved. In addition it will contain the general abstract concepts, like *topic*, *association* etc. The organizations involved must come together in advance and elaborate a common ontology.

The next step is to merge the different topic maps across the domains. Since the *Ambulance* topic maps and the *Police* topic maps share the same ontology and are using the same PSIs, they can merge. To be able to share information across domains is therefore made possible. In the MANET, the personnel can import and export topic maps when required. The merging should be an automated action taken care of by the application. The personnel should not have to know where the information is gathered. This leads to the fact that a fireman can have access to data in the ambulance topic map. For hiding sensitive data, this will be taken care of by typing.

Another advantage with merging topic maps is that the user can be equipped with just a small part of the topic map in front to spare resources on the limited mobile device. Later when the user needs another part of the same topic map, the application will fetch the necessary parts. This way the resource-weak device can handle topic maps of larger size. The unused parts of the topic map can be removed by the topic map engine.

We have not seen the crucial need to store the topic map. The topic maps are elaborated up front and should not be a need for storing them. If storage is needed, the topic maps can be transferred to the laptop at the end of the operation. It could be necessary to store information about the injuries.

To be able to take full advantage of Topic Maps, it is important to be prepared to a certain extent. Before an accident occurs, the emergency staff cannot predict which information will be essential. Every emergency is different and the course of the accident will change from time to time. By using experience and expertise, the emergency personnel can predict some of the knowledge needed in different scenarios. If possible, it could be practical to merge the desired topic maps in advance. If the accident happens in a tunnel and a certain train type is involved, they can load the necessary topic maps e.g. about trains and merge them.

One of the benefits of the Topic Maps standard and knowledge integration is the vision of the seamless knowledge. The Topic Map standard has its own protocol for exchanging topic maps across the network, the TMRAP [62]. The merging of topic maps can be one of the most futuristic and beneficial mechanisms for large distributed enterprises.

To gain overview of a large accident can be very valuable. In the emergency example, the number of people being involved can be numerous and the damages likewise. The possibility to gain an overview over the emergency scenario as quickly as possible can contribute greatly to the coordination work and reduce the time spent on coordination verbally. The bracelet can be very valuable for the medical crew when they arrive at the scene. They can quickly be informed of the number of people who are critically injured and give them the necessary treatment first. In addition, an overview of the wounded can save a lot of time. The personnel located in the MANET can update each other without having to inquire the EO. The EO can download the findings and registrations from the PDAs. The use of MANET requires minimal configuration and quick deployment.

5 Merging

This chapter is based on the Topic Map Data Model. Merging is one of the cornerstones of the Topic Map paradigm. The mechanism can enable sharing and exchange of knowledge in distributed systems. Merging makes it possible to take two arbitrary topic maps and merge them into one. The merging operation removes all the redundant topic map constructs if two topics represent the same subject. This makes the merged topic grow richer by inheriting the characteristics from another topic. This function can enable reuse of knowledge across organizational boundaries.

Merging two topic maps

The focus of the thesis is to merge two topic maps. The different organizations at an emergency scene shall be able to merge their topic map with a topic map from another organization. Figure 31 illustrates the merging between two topic maps. The merged topics and associations are coloured green.

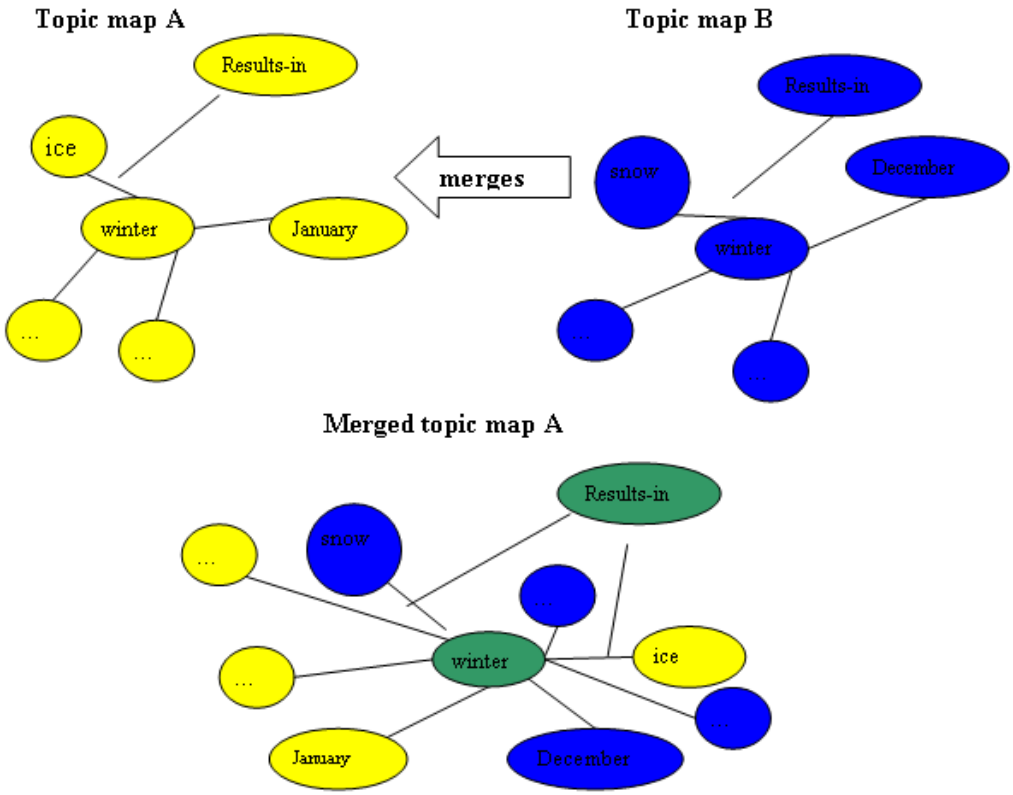


Figure 31 Merging

When merging topic map *A* with topic map *B*, the topic map A will be altered and expanded with the topic map items from topic map B. Topic map B is left unchanged.

A set of *equality rules* establish the equality of two topic map items. When two topics are defined as equal, they must merge according to the *merging rules*. The merging rules are procedures for performing the actual merging of two topic map items.

5.1 Equality Rules

To decide if two items should merge or not, the merging concept is based on certain equality rules. The merging criteria are divided into two categories. The first category is the subject-based merging. As the name indicates, the rule decides the equality of two topics based on the subject identity they inhabit. Two topics represent the same subject if *one* of the conditions listed below are true,

1. At least one equal PSI, or
2. At least one equal item identifier, or
3. At least one equal subject locator, or
4. The item identifier of the one topic is identical to the PSI of the other

When two topics have the same item identifier they do not necessary represent the same subject. When topic maps are serialized (converted to XTM format) the last part of the item identifier is converted to the *id* attribute of a topic element. These ids in XML-based documents should always be unique. These *ids* are used to refer to other topics. To avoid ambiguity topics within the same topic map cannot have the same ids.

The other category is name-based merging, in accordance with the Topic Name Constraint (TNC),

5. If two topics have equal names in the same scope, they represent the same subject and hence must be merged

This constraint is controversial. Since topic naming is a subjective process performed by humans, it could lead to the fact that two topics *accidentally* have the same name. If two topics in addition, have the same scope, they must merge regardless of the fact that they could have different subject identity. The only exception is, two topics with different subject locators, they shall *never* be merged.

However, if the TNC is not applied the topic map could contain topics with equal names in the same scope, so-called homonyms. They do not represent the same subject, so according to the subject-based equality rules, it is correct. The topics are displayed by using topic name, and two equal names in the same scope could confuse the user. In consistency with item identifiers, topics should not have equal names in the same scope in the same topic map. The TMAPI has a feature for indicating the support of the TNC, called *mergeByTopicName*. This feature must be implemented and the application will run according to the settings in this feature. This is up to the application developer to decide. The feature is a string in a file that is set to true or false, the topic map engine will read this file at start-up.

Equal reified items

Another criteria that defines equality is,

6. The same information item in their reified properties

If two topics have reified the same topic, they are representing the same topic and hence the same subject. They must merge.

1. Equal PSI
The usage of PSIs is a great contributor to merge correctly. When subjects are non-addressable, the identity of the topics should be defined by using a PSI. There is already a set of very general PSIs and new sets are under development. When elaborating a topic map, it will be necessary to create new PSIs that describe the domain of matter. PSIs indicate the subject the topic represents by supplying a URI to the textual description (http://psi.ontopia.net/psis/#type). If two topics have at least one equal PSI, they represent the same subject, and must merge.

It is possible for a topic to have several PSIs, since different topic map applications can use different sets of elaborated PSIs. To enable merging between companies that use different PSIs, they can exchange PSIs and use them in addition to their own PSIs. This can ensure a more precise merging. If the different topic maps use a different set of PSIs, the merging can be of little use. The example below demonstrates a topic with two overlapping PSIs from different organizations.

```
<topic id="rescuetopic">
    <subjectIdentity>
       <subjectIndicatorRef xlink:href="http://www.policePSI.org/#rescue"/>
       <subjectIndicatorRef xlink:href="http://www.firePSI.com/#rescue"/>
    </subjectIdentity>
</topic>
```

2. Equal item identifier
As explained in *2.3.5.1*, the item identifier is a concatenation of the topic map locator and the id attribute of the topic in the XTM document. The item identifier is created when the topic map is deserialized. If two topics in the same topic map have the same id, they should therefore merge. This should be checked at deserialization and an automatic merge should be performed according to the data model.

```
<topic id="rescuetopic">
    <subjectIdentity>
       <subjectIndicatorRef xlink:href="http://www.psi.org/#rescue"/>
    </subjectIdentity>
</topic>

<topic id="rescuetopic">
    <baseName>
        <baseNameString>rescue<baseNameString>
   </baseName>
</topic>
```

These topics inside the same topic map will merge and result in:

```
<topic id="rescuetopic">
   <subjectIdentity>
        <subjectIndicatorRef xlink:href="http://www.psi.org/#rescue"/>
    </subjectIdentity>
    <baseName>
        <baseNameString>rescue<baseNameString>
    </baseName>
</topic>
```

In the emergency use case, the organizations will have unique topic map locations elaborated in forehand. This way, the item identifiers cannot be equal if they belong to topics from *different* topic maps. This prevents the case, where two topics possess the same id but do not represent the same subject.

In a more general sense, the users of the system could provide arbitrary names as unique topic map ids instead of the documents address. This would lead to a higher possibility for an equality match. In general, one cannot presume that item identifiers from different topic maps are unequal. A correct implementation of the item identifier is highly recommended, and in addition the usage of document paths as ids. This will help to ensure a correct merging process.

3. Equal subject locator
If two topics represent the same addressable subject, they will have the same value in their subject locator property. The subject locator is the address to an electronic resource that the topic represents.

```
<topic id="rescue_standard">
   <subjectIdentity>
      <resourceRef xlink:href="http://www.rescue.org/rescuestandard.pdf"/>
    </subjectIdentity>
    <baseName>
        <baseNameString>PDF document about rescue standard<baseNameString>
     </baseName>
  </topic>

<topic id="rescue_standard ">
   <subjectIdentity>
      <resourceRef xlink:href="http://www.rescue.org/rescuestandard.pdf"/>
    </subjectIdentity>
    <baseName>
        <baseNameString>rescue standard<baseNameString>
    </baseName>
  </topic>
```

The topics are merged into one.

```
<topic id="rescue_standard">
   <subjectIdentity>
      <resourceRef xlink:href="http://www.rescue.org/rescuestandard.pdf"/>
   </subjectIdentity>
   <baseName>
        <baseNameString>PDF document about rescue standard<baseNameString>
   </baseName>
   <baseName>
        <baseNameString>rescue standard<baseNameString>
   </baseName>
</topic>
```

4. Equal item identifier and PSI
If the PSI is a reference to another topic, they should merge. This is not encouraged, and considered as bad topic mapping. As the example shows, the *rescue_standard* topic *is* the *rescuetopic*.

Topic A:

```
<topic id="rescue_standard">
   <subjectIdentity>
        <subjectIndicatorRef xlink:href="http://www.psi.org/#rescue/>
    </subjectIdentity>
  </topic>
```

Topic B:

```
<topic id=" http://www.psi.org/#rescue ">
        ...
</topic>
```

The merged topic will look like this:

```
<topic id="rescue_standard">
    <subjectIdentity>
        <subjectIndicatorRef xlink:href="http://www.psi.org/#rescue/>
    </subjectIdentity>
</topic>
```

Deciding which item identifier to use

When a topic merges with another it will have two item identifiers, one from topic B and the one already existing. This could sound like a topic identity problem. There are no general rules on this matter, concerning the creation of a topic map engine. However, in the MTV, topics are identified by the element id they possess in the file system record and *not* the item identifier. The item identifier is merely used as an indicator of the origin of the parent topic map. By extracting the topic map locator from the item identifier string, you can get information of the origin of the topic. The implementation of the serializer decides if both item identifiers should be used or if they should be rejected. A suggested solution is to keep the original item identifier of topic A to recreate the id attribute in the XTM.

5. Equal topic name in the same scope

If two topics have the same name in the same scope they are defined as equal and should be merged. If they do not merge, it will be difficult to display the topics and manage to differentiate them. According to the example below, they must merge.

```
<topic id="espa_train_accident">
    <baseName>
         <scope>english</scope>
         <baseNameString>The Espa train accident<baseNameString>
    </baseName>
    <instanceOf>
         <topicRef xlink: href="#accident"/>
    </instanceOf>
</topic>
...
<topic id="espa_accident">
    <baseName>
        <scope>english</scope>
        <baseNameString>The Espa train accident<baseNameString>
        <variant>
            <scope>
                  <topicRef xlink: href="#sort"/>
             </scope>
           <variantName>
```

```
                <resourceData>Espa, train accident</resourceData>
            </variantName>
        </variant>
    </baseName>
</topic>
```

The result will be:

```
<topic id="espa_train_accident">
    <instanceOf>
        <topicRef xlink: href="#accident"/>
    </instanceOf>
    <baseName>
        <scope>english</scope>
        <baseNameString>The Espa train accident<baseNameString>
        <variant>
            <scope>
                <topicRef xlink: href="#sort"/>
            </scope>
            <variantName>
                <resourceData>Espa, train accident</resourceData>
            </variantName>
        </variant>
    </baseName>
</topic>
```

The item identifier is added to the merged topic, but this is not visible in the XTM 1.0 version.

6. Equal information item in their reified properties

This last criterion involves reification as elaborated earlier in the 2.3.5.4 section. A topic can reify a topic map item such as topic names, associations and occurrences, to be able to assign characteristics to them.

The reified:

```
<topic id="espa_train_accident">
    <! - The topic name is reified by the espa_accident_name topic -->
    <baseName>
        <topicRef xlink: href="#espa_accident_name"/>
    </baseName>
</topic>
```

The reifier:

```
<topic id="espa_accident_name">
    <baseName>
        <baseNameString>The Espa train accident<baseNameString>
    </baseName>
</topic>
```

The reified topic has a so-called [reifier] property, which contains the reifier (topic item). When a topic has reified e.g. a topic name, this information item is added into the [reified] property of the topic. These properties are created during deserialization of a given topic map. When two topic maps are merged, topics with equal information items in their [reified] properties are indeed equal and must merge.

5.2 Merging Rules

This section is based on the data model where nothing else is given. Some of the details of the merging rules are accustomed to the technical implementation. However, this does not collide with the essentials in the merging process described in the data model.

When two topics are identified as equal the process of merging starts. Topic A topic inherits the union of the characteristics of the two equal topics. To be able to merge the characteristics of each topic, it is necessary to decide if the properties are equal. The merging rules sketch out the procedure for merging the different topic map items. The properties are topic names, variants, occurrences and association roles. There are rules for deciding the equality of each of these properties. If they are equal, there is no need for adding the property from topic B. According to the example below, the type of topic *A* must be compared to the type of topic *B*. Then a completely new topic equality check is done. The topic *accident* must be compared with the topic *acc* in topic map tmB. The subject identity must be established and if it is defined as equal, the type is not added to topic A, since the topic already is of the same type. This avoids redundant information items in the topic map.

If the properties are unequal, the property from topic B will be added. The depth of the merging process is up to the application/engine developer. There will be a trade-off between correctness and speed. The use case can put constraints on the elaboration of the depth of the algorithm.

When two topics are identified as representing the same subject, the merging rules decide whether the characteristics are equal. For every characteristic in topic *A* that is not found in topic *B*, this is added to topic A. In the syntax example below, topic *B* and *A* are identified as equal by their item identifier. To decide whether the topic type should be added to topic A, a new round of equality checking between the types of the two topics must take place. To establish the equality of the topic types, the next step is to establish the subject identity of the *accident* topic and the *acc* topic. By using the equality rules, equal PSIs are detected and the topics are declared equal. This requires no action.

TOPIC MAP tmA (includer topic map)

Topic A

```
<topic id="espa_train_accident">
    <!—is of type accident - ->
    <instanceOf>
        <topicRef xlink: href="#accident"/>
    </instanceOf>
 </topic>

<topic id="accident">
   <subjectIdentity>
       <!—the PSI - ->
      <subjectIndicatorRef xlink:href="http://www.psi.org/#accident"/>
    </subjectIdentity>
</topic>
```

TOPIC MAP tmB (included topic map)

Topic B

```
<topic id="espa_train_accident">
    <!—is of type acc - ->
    <instanceOf>
```

```
        <topicRef xlink: href="#acc"/>
    </instanceOf>
 </topic>

<topic id="acc">
     <subjectIdentity>
        <!—the PSI - ->
        <subjectIndicatorRef xlink:href="http://www.psi.org/#accident"/>
    </subjectIdentity>
</topic>
```

Creation of new topic or not?

In the merging description of the data model, a new topic C is created every time two topics merge. The merging rules say that a new topic C should inherit the union of the characteristics from the two merging topics. This would create a new element id in the file system, and the item identifiers of A and B are added. Everywhere that topic A and B is referenced in the topic map must be replaced by C. This could be a complex and/or time consuming process. After the process of merging is completed, topic A is removed from topic map tmA. In my implementation I omitted the creation of a new topic C, and added the characteristics of topic B to topic A. this way the algorithm do not have to add the characteristics of topic A to the new topic C. In addition, there is no need for a delete process on topic A. I believe this is a good solution.

Merging topics

The following information in topic B should be considered when adding characteristics to topic A. the items are divided into two categories, one contains the items that must be merged and the other category contains the items that are added after a duplicate test is performed.

Must merge:

- [occurrence(s)]- Every occurrence that topic B may have, must be assigned to topic A.
- [topic names] –in addition to their [variants]
- [roles played in associations] – If topic B is a player in an association role, the reference to the association role must be added to topic A to maintain the information.

Add if the item is not already contained in A,

- [PSIs]
- [subject locators]
- [item identifiers]
- The type(s) – If topic B is typed with topic D, topic A must also be typed with the same topic type(s). This is done by creating a super-subtype association between A and the topic that corresponds to D2 in topic map tmA.
- *[reified]*

However, adding these information items and relations to other items can lead to redundant information items in topic A. Topic A may already contain the same topic name or the same PSI. To avoid duplicates the information items must also merge. This is a similar procedure to the merging of the topic. Nevertheless, the properties in information items like, topic name, and occurrence and so on are different from the properties of a topic and hence require a different merging approach.

If the [reifier] property of topic A is null, and the [reifier property] of topic B is not null, the [reifier] is added to topic A. If the topics have information items in their [reifier] properties, these information items must merge.

Merging topic names
The values in the following properties must be equal if they should be merged,

- [name]
- [type]
- [scope]
- [parent topic]-the topic it belongs to

If both topic A and topic B has null value in their [type] or [scope] properties, this is considered equal. If these properties are equal, the topic names are merged. Topic name A inherits the [item identifier] and [variant names] from topic name B.

Merging variants
The values in the following properties must be equal if they should be merged,

- [name]
- [data type]- it can be a reference to another topic or a string value
- [scope]
- [parent topic]-the topic it belongs to

If both topic A and topic B has null value in their [type] or [scope] properties, this is considered equal. If these properties are equal, the variant names are merged. Variant name A inherits the [item identifier] from variant name B.

Merging occurrences
The values in the following properties must be equal if they should be merged,

- [value]-the URI
- [scope]
- [type]
- [parent topic]-the topic it belongs to

If both topic A and topic B has null value in their [type] or [scope] properties, this is considered equal. If these properties are equal, the occurrences are merged. Occurrence A inherits the [item identifier] from occurrence B.

Merging associations

To *merge* associations, the association [type], [scope] and the [association roles] must be identical. The elements that are merged are the *[reifier]* properties and the item identifiers.

If association A and association B has null value in their [type] or [scope] properties, this is considered equal. If these three properties are equal, the association are merged. association A inherits the [item identifier] from association B.

To decide if two associations are equal the association roles must be compared. The values in the following properties must be equal if they should be merged,

- [role/type]-the topic that is used as role
- [player]- the topic that is the player of the role
- [scope]

To check the equality of the [player] and [role] properties we must apply the equality rules for topic items.

The level of correctness
According to the merging rules, there should not be any redundant information items in a topic map. However, the data model points out at that the rules do not ensure this 100%. There are several reasons for this. If the developers are not using PSIs and the topic name constraint is not used, there are not many ways of deciding the equality of two subjects. In addition, the merging process can be very deep, of performed correctly in all aspects. The solution is to decide where to stop, depending on the application range. It is up to the application developer to decide where to stop. The type of a topic is also a topic that could have a type which has a type and so on. To check equality of each and every nested topic reference is time consuming and resource demanding.

Triggering merging

Merging can be automatically applied during deserialization of the XTM document. By deserialization, I mean the process of building an instance of an implementation's internal representation of the data model from an instance of topic map syntax. There are three ways to trigger a merging,

1. By explicit application programming, the application can decide to merge two selected topic maps.
2. By entering a <mergeMap> element in the XTM document, this element points at a topic map that should be merged
3. When the topic maps engine processes the <topicRef> element in the topic map, this could be a reference to a topic in another topic map. If so, the engine should retrieve the entire referred topic map and merge it with the current topic map.

The mergeMap directive can be used to automate the merging of two topic maps. This could be used when the application knows that the two topic maps should be merged. If the mergeMap directive should be used in the use case scenario, the rescue personnel must know in forehand what information can come in use or needed. This would not be flexible enough. During an emergency the needs and actions required cannot be planned, therefore, the personnel will have the ability to import and merge topic maps when they require it. The third case ensures that a topic map imports external topics

Lack of merging applications
As discovered in the 2.3.5, there are not many examples of applications that merge topic maps in distributed environments. Merging is based on being able to establish subject identity by utilizing PSIs. Perhaps the development of a common set of PSIs is the problem. The OASIS has standardized the use and development of PSIs, and a set of PSIs for geography and language has been elaborated. Nevertheless, the road is long. It is impossible to create a PSI for every subject of the real world. Even to create a set belonging to a domain ontology can be a challenge.

Another issue is the complexity of the algorithm. The implementation can be very detailed and “deep” and the trick is to know where to stop. The procedure for merging is explained in a very superficial manner in the data model and in the ISO specification. This can open up for many interpretations. This can lead to many different and maybe flawed solutions. This is not beneficial to the Topic Map community. The merging procedure does not ensure a 100% precise result. The merging process cannot promise to remove every redundant item. Maybe the procedure should be standardised in a more detailed manner, perhaps a pseudocode could be elaborated. In addition, an example implementation could be demonstrated.

6 Implementation of Merging

6.1 Introduction

Samuel Vigdal has already performed some benchmark testing with loading and querying topic maps on two different mobile phones in his master thesis [6]. His results showed that both loading and querying was possible within reasonable time. In addition, the size of the mobile screen was large enough to read the information needed. The mobile phones he used for testing had different memory and processor speed. The oldest mobile phone used long response time on the largest topic maps, the newer model performed within 2-3 seconds. Merging is a quite resource demanding process compared to the queries he performed. However, the mobile phone speed is improving and new models are constantly under development. Therefore, although some of his tests had longer response time, the constant development within mobile devices left little room for doubt about the usage of resource-weak devices for merging. This chapter introduces the data structures and storage system in the MTV. Since tinyTIM was the study case before the implementation I introduce the basics of the engine. Finally yet importantly, my implementation of merging in the MTV is elaborated and discussed.

6.2 Mobile Topic Viewer

The MTV was a continuation of the Topic Map engine in the SHARK project (see Chapter 3). It was further developed by master student Marcus Walla [59] at the University of Berlin last year. The supervisor and idea-maker was Thomas Schwotzer. This engine is the first of its kind, namely a Topic Map engine for mobile devices, implemented with the J2ME API. Keywords in such an implementation are speed and size. A mobile device is resource weak and limited on both memory and storage space. The sizes of the topic maps are limited. The mobile device is dependent on a fairly small and lightweight midlet to be able to run in and deliver results to the user in due time.

The implementation is TMAPI specific and contains some extra packages for handling file system and implementation of Java ME specific data structures. The MTV stores data by using the Record Management System (RMS) provided by Java ME. It can also operate without using persistence. MTV supports serialization and deserialization of XTM files. MTV can parse XTM 1.0 compliant files. For faster access, Walla has made a file system, which serves as a cache.

The filesystem implementation
To ensure a fast midlet Walla [5] implemented a filesystem for temporary storage of topic maps. The *RecordStore* in Java ME is responsible for persisting data. Reading and writing to a topic map that is stored in the *RecordStore* takes resources and time. To solve this problem, a filesystem serves as cache between the memory and the persistent storage. It is possible to read the topic map into the filesystem without storing it. Whenever a topic map is created the engine reads from the *RecordStore* and checks if the topic map already exists. In addition, every topic map that is stored in the *RecordStore* is loaded into memory. To be able to handle several topic maps in the file system concurrently, a unique locator must be provided for each.

The filesystem uses a mapping table to define which element in the file system corresponds to which topic map element. The topic name element could e.g. have placement 20 in the filesystem.

The data structures
The data structure of the topic map system is implemented using AVLTrees for storing topic maps are stored. A new topic map is stored using a table where every record in the table is stored as an AVLTree. The AVLTree is a balanced tree structure and hence will ensure a faster indexing [63].

When creating the topic map, six AVLTree objects are instantiated and put into the table. Each item, topics, associations, subject locators and so on are stored in new AVLTrees. Each internal tree has implemented a comparing class to compare an integer or a string. Figure 33 illustrates that there are only indexes implemented for topics and associations.

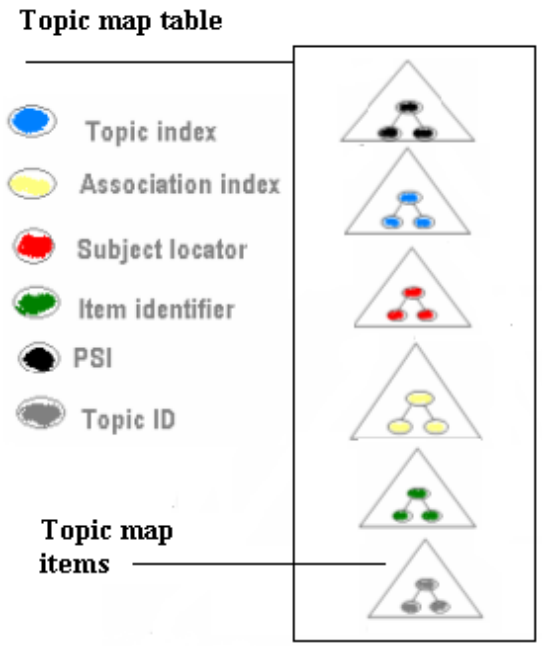


Figure 32 Topic map data structure

According to the data model, a topic that refers to a topic in another topic map should be merged into the current topic map. This is not implemented in the engine yet. Therefore, if topic map *tmB* is removed, the reference to the topic in topic map *tmB* will be lost. In addition, if *tmA* is serialized, the connection to *tmB* will be invalid. When MTV searches for a topic, it searches within every topic map in the topic map system. This way, the user will not detect if a topic is fetched from another topic map. I believe that the search should be limited to the current topic map. If a topic references a topic in another topic map, an error should be thrown.

Possible improvements
After studying the mobile engine there are some shortcomings that should be mentioned. The MTV is not a finished product and there is room for improvements. There are concepts in the Topic Map standard that is not implemented in the MTV. Two important mechanisms are the merging of arbitrary topic maps and the reification mechanism. The list below is seen in perspective with the use case of this thesis. The engine is a high-quality implementation of the TMAPI, and the file system solution makes the engine faster and less resource demanding.

This lists the possible improvements; each item is discussed further down,

1. Merging
2. Implementing indexes
3. Reification
4. Remove
5. XTM 2.0 compliant
6. Data model compliant

1. Merging
The merging functionality was not implemented. To be able to merge topic maps, the mobile engine must depend on other devices to perform the merging. Since there is no other topic map engine developed for mobile devices, this is bound to be a laptop or a stationary computer. Topic map engines that perform merging are e.g. tinyTIM, TM4J and the OKS engine.

2. Indexing
The TMAPI contains a set of interfaces for indexing the information items in the topic map faster. MTV has an implementation of the interface for indexing the *topic* items. The TopicIndex gives access to topics through the [type], [subject locator] or the [PSI]. According to Samuel Vigdal [6], the implementation of the topic indexes improved response time considerably when querying the topic map. Indexes for associations, topic names, occurrences, variant names and association roles are not implemented.

3. Reification
Reification is not implemented.

4. Remove
When trying to remove topic maps or topics an exception occurs. I believe this is due to the use of the persistent storage, when this is not always used.

5. XTM 2.0 compliant
The XTM 2.0 version has improved the syntax naming and changed the DTD format. These changes are improvements and a conversion to the syntax is not a huge task. I have already written comment where the code must be altered to adjust to the new version.

6. Data model compliant
The data model has some changes that are reflected in the syntax of the XTM 2.0. A correct implementation of the item identifier is one of the most important tasks.

6.3 Java ME

To test the most resource-weak devices the choice fell on mobile phones. Most mobile phones have, today, support for Java. The mobile technology Stack is based on the Java Platform, Micro edition (Java ME) [63], specification and consists of the Virtual Machine and CLDC libraries as the foundation and the Mobile Information Device Profile and optional packages on top. To avoid time-consuming testing and loading on real mobile devices, the Wireless Toolkit (WTK) for Java ME offers emulation. A midlet can be run directly from the toolkit or the Java ME ca be plugged-into a java editor and run from there. The emulator has four different mobile profiles for testing [58].

6.4 The Actual Implementation

6.4.1 Overview

The implementation of the merging algorithm requires a thorough understanding of the Topic Map Data Model, syntax and TMAPI. I will try to give a detailed picture of the process without making it too complex. I will leave out the parts I believe are too detailed or unnecessary for the general understanding. The Appendix contains the Java implementation of the merging process.

The implementation consists of a method for copying the topics and associations in a topic map into another topic map by merging. This method makes use of a merging method on topic level and a method for adding or merging associations. Since the MTV is based on the XTM 1.0, the algorithm is also based on this [29].

In addition, the name-based equality rule (*mergeByTopicName* see 5.1) can be applied or not. If the algorithm skips the topic name comparisons, many loops can be avoided. This is a feature in the TMAPI that can be switched on or off. This leaves some choices,

Merge two topic maps,

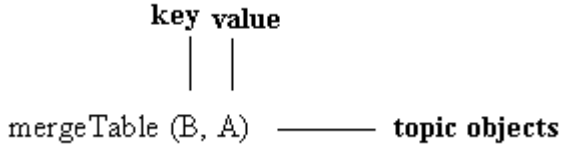
- complete merge
  - with mergeByTopicName feature on
  - with mergeByTopicName feature off
- limited merge
  - with mergeByTopicName feature on
  - with mergeByTopicName feature off

6.4.2 Merging Topic Maps

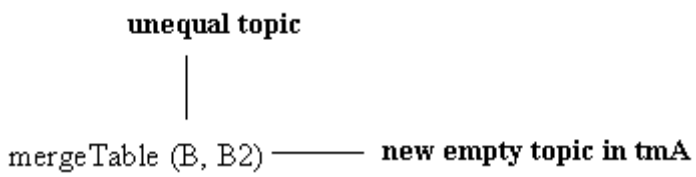
The TMAPI contains two interface methods that should be implemented in a topic map engine. The method on the topic map level performs equality tests to decide whether the topics from the included topic map should be merged or added. Every topic from the included topic map is compared it with every topic from the includer topic map. If the equality test gives a match, the two topics are merged.

Topic map **tmA** (Includer topic map)
Contains topic A
Topic map **tmB** (Included topic map)
Contains topic B

Topic map tmA shall merge with tmB, tmA is expanded, and tmB is left unchanged. If topic A and B are defined as equal they are inserted into mergeTable as a merging pair. Topic B is used as key and topic A is the corresponding value.



If topic B does not match with any other topic in tmA a new (empty) topic item, B2 is created in tmA. B and B2 are inserted into the mergeTable as described in the previous section. This way, every topic from tmB is *merged* with another topic from tmA.



Topic B is merged with either an “empty” topic or a “real” topic from tmA.

When all the topics from tmB have been tested for equality, every topic pair in the *mergeTable* is merged. When merging two topics, every property of topic B (the key) is compared to the similar set of properties in topic A (value). If topic A has an empty set, the set from topic B is added. The objective is to avoid adding equal characteristics that already exist in the topic. This process is a challenge regarding keeping the correct references and checking equality of each and every property of the topics.
After the topics have merged, the associations are processed.

Hashtables have been used for storing elements during the merging process. The possibility of storing a key-value pair is a good solution when indicating that two topics are in fact the same. This way, it is possible by using the key, to fetch the topic (value) in tmA that will represent topic B.

6.4.2.1 Algorithm 1-The Equality Tests

The algorithm for merging topic maps is sketched out in pseudocode below. The different stages in the merging process are marked with a number and will be discussed in more detail in the next section.

Algorithm mergeIn (TopicMap includedTM):

```
{mergeTable: = nonmergeTable: = new HashTable;
limited: = getMergeVersion; mergeByTopicName := getMergeByTNFlag;
For each t2 in otherTM do
   merge: = false; diffSubjLoc := false;
   For each t1 in thisTM and while !merge do
        1. for each SL2 in t2.subjectLocators do
                for each SL1 in t1.subjectLocators do
                   If SL1=SL2 then merge: = true and go to 7;
                   else diffSubjLoc:= true
                   fi;
                od
          od;
          if diffSubjLoc=true go to 7;
        2. for each PSI2 in t2. publishedSubjectIdentifiers and while !merge do
                for each PSI1 in t1. publishedSubjectIdentifiers do
                   If PSI1= PSI2 then merge: = true;
                od
          od;
        3. for each II2 in t2. itemIdentifiers and while !merge do
                for each II1 in t1. itemIdentifiers do
                    If II1= II2 then merge: = true;
                od
         od;
        4. for each PSI2 in t2.publishedSubjectIdentifiers and while !merge do
                for each II1 in t1.itemIdentifiers do
                    If II1= PSI2 then merge: = true;
                od
         od;
        5. for each II2 in t2.itemIdentifiers and while !merge do
                for each PSI1 in t1.publishedSubjectIdentifiers do
                    If II1= PSI2 then merge: = true;
                od
         od;

        6. If mergeByTopicName
             for each TN2 in t2.topicNames and while !merge do
                for each TN1 in t1.topicNames do
                    If TN1 = TN2 then merge: = true;
                od
             od;
           fi;
    od;
   7.
   If merge then
        mergeTable.add (t2, t1)
     else
        nonmergeTable.add (t2)
     fi;
od;
for each t in nonmergeTable do
        t :=thisTM.createTopic
        nonmergeTable.add (t, t2)
od;
for each key in mergeTable do
        m2:= key;
        while mergeTable(key) do
              m1:= mergeTable (key);
              m1.mergeIn (m2, false, mergeTable);
        od;
od;
if(limited) addAssociations();
else mergeAssociations();}
```

The following sections elaborates the different steps in the previous algorithm. To avoid unnecessary iterations in the code, I have placed the equality tests in a certain order. The tests that are most likely to be true are performed first. Whenever a test has a positive result, a variable is set to true and the rest of the algorithm is ignored. This solution avoids unnecessary looping.

Order of sequence:
1. Subject locator
2. PSI
3. item identifier
4. item identifier vs. PSI
5. PSI vs. item identifier

If mergeByTopicName is enabled,
6. topic name

1. Subject locator
The subject locator is the address of the subject the topic represents. It is incorrect to merge two topics with different subject locators, since this means that they are representing different subjects. Therefore, this property is compared first. If the topics have values in their [subject locators] property, there is no reason to run through the rest of the equality tests. If the properties are equal, merging is true and if they are not, merging is false. The code jumps out of the for-loop and the topic is added to the mergeTable or nonmergeTable directly. This leaves out many unnecessary loops.

2. PSI
The PSI is an alternative to the subject locator and a way of establishing the identity of the topic. The PSI is one of the main reasons for making subject-based merging possible. Therefore, if two topics have equal PSIs they are reifying the same subject. A topic can have several PSIs and must therefore compare every PSI of topic B with every PSI in topic A. If one PSI matches another PSI, a merging must happen. PSIs are elaborated in 2.3.5.1.

3/4/5 Item identifier
The item identifier is a concatenation of the topic map locator and the id attribute in the XTM file. If two topic maps have the same locator, (a random string could have been used instead of the unique document path) the chances for having equal item identifiers in different topic maps is bigger. Overall, the item identifiers of two topics in *different* topic maps are almost never bound to be equal.

This statement presupposes a correct implementation of the topic map locator (document address path) and the item identifier. As mentioned in chapter 5, a developer should never assume that the users utilize the topic map engine as intended. Therefore, it is crucial that the application use document paths as locators for the topic maps. This can guarantee that the topic maps will have different locators and hence have different item identifiers.

6. Topic names
If the *mergeByTopicName* feature is enabled this test is performed. Otherwise, this step can be skipped. A topic name must have the same [value] and the same scope to be equal. The value property is the string representation of the name and the scope is the context that the name is

valid in. If the topic names have a null- value in their [scope] property, the unconstrained scope is used, and the scopes are considered equal.

6.4.2.2 Algorithm 2-Merging Topics

Below follows the pseudocode for merging topics. The method merges the topics in the mergeTable after the equality tests have been performed. Each property in t2 from the included topic map is compared with the corresponding properties in *this* topic. *This* topic is expanded with properties from t2. The parameters in the method contain the topic to be included (t2), a flag for indicating an internal merge, and the *mergeTable* reference for lookup. Modify parameter is always set to false in t

```
Algorithm mergeIn (Topic t2, boolean modify, Hashtable mergeTable) {
   mergeByTopicName: = getMergeByTNFlag;
1. SL2:= t2.subjectLocator; SL1:= this.subjectLocator
   If SL1 ≠ null do
       If  SL1 ≠ SL2
            throw exception
   else this. add(SL2)
   fi;

2. for each RP1 in t2.getRolesPlayed
     this. add (RP1);
    od;

3. for each TYPE1 in t2. types do
     add : = true
     topicRef = mergeTable.get(TYPE1)
     for each TYPE2 in this. types and add= true do

        If TYPE1= topicRef then add: = false;
     od;
     if add then
            this. add(topicRef);
     fi;
  od;

4. for each II2 in t2. itemIdentifiers do
     add : = true
     for each II1 in this. itemIdentifiers and add = true do
            If II1= II2 then add = false;
     od;
     if add then this. add(II2):
 od;

5. for each PSI2 in t2.publishedSubjectIdentifiers do
     for each PSI1 in this. publishedSubjectIdentifiers do
            If PSI2 = PSI1 then add = false;
     od;
     if add then this. add(PSI2)
 od;

6. if mergeByTopicName
     for each TN2 in t2.topicNames
          merge = false;
          for each TN1 in this.topicNames and merge = false do
               if TN1 = TN2
                    If TN1.getScopes = TN2.getScopes then merge= false;
                    fi;
```

```
                fi;
            od;
            If merge then TN1.add(TN2.getVariants)
            else
                new topicName(TN2.getName,TN2.getScope,TN2.getVariants)
                this. add(topicName);
            fi;
        od;
    fi;

7.  for each OCC2 in t2.occurrences do
        add: = true;
        for each OCC1 in this. occurrences and add = true do
                If OCC2 = OCC1 then add = false;                                    fi;
        od;
        if add then this. add(OCC2)
    od;
if modify then t2.remove

}
```

1. Subject locator
If the [subject locator] properties are unequal, an exception should be thrown. This is thoroughly checked in the previous method. The test is not implemented with support for multiple subject locators. I consider multiple subject identifiers as a seldom case.

2. Association roles
If the topic has values in the [association roles] property, these must be added. I have not compared the association roles between the two topics. Therefore, this step could lead to duplicates. An equality check would be very “deep” and require much iteration, especially if the topics have many association roles.

3. Types
The [types] are tested for equality by using the mergeTable as lookup. If a type in t2 is not contained in the properties of this, the reference is added.

4. Item identifiers
The [item identifiers] property is added if they are unequal.

5. PSIs
The [PSIs] are added if they are unequal.

6. Topic name
If mergeByTopicName is enabled, the names and their belonging scopes are compared. If the topic names are considered equal, the variants of the topic name are added to the topic name of *this*. Scopes are also topics, so the mergeTable can be used to perform a lookup and compare the two scopes

7. Occurrences
Every value in the [occurrence] properties are compared, if *this* has no matching pair the occurrence is added.

6.4.2.3 Algorithm 3-Merging Associations

The pseudocode below has been simplified to make it legible. The use of mergeTable for fetching the correct reference is demonstrated. The algorithm loops through every association from the otherTM and compares [type] and [associations roles] with every association from thisTM. As soon as the association has found a match, the loop is terminated and the [item identifier] is added to the equal association. If an association do not match any other association in the topic map, a new association I created with the same properties as the current one.

```
mergeAssociations (TopicMap otherTM, Hashtable mergeTable) {
   For each ASSOC2 in otherTM do
        duplicate: = false; copy := true;

         For each ASSOC in thisTM and copy = true
                  topicRef = mergeTable.get(ASSOC2.getType)
                  If ASSOC.getType = topicRef then duplicate = true
                  If duplicate
                      topicRef = mergeTable.get(ASSOC2.getAssocRole)
                      If ASSOC.getAssocRoles = topicRef
                            then copy = false
                      else duplicate = false
        if copy
          topicRef1 = mergeTable.get(ASSOC2.getType)
          //this is simplified
          topicRef2 = mergeTable.get(ASSOC2. getAssocRoles)
          create new association(topicRef1, topicRef2)
        else
                  ASSOC. add (ASSOC2.getItemIdentifiers)

}
```

Pruning the algorithm

I customized the implementation to some degree to meet the requirement concerning a fast system time. I made it possible to choose a limited merging version. This version can be applied if the time aspect is more important then the level of preciseness. To create a limited version I added some shortcuts and omitted some of the tests that I defined as less likely to be true. The MTV has a properties file for defining some values. These are e.g. topic map size and topic map element size. I inserted a property in this file to indicate the merging version used. The MTV reads this file at start-up , this way the merging algorithm knows which version to choose, either the limited or the complete one.

Equality tests, limited merge

1. Subject locator
2. PSI

If *mergeByTopicName* feature is on,

3. Topic name

To achieve a faster merging process it is possible to omit the item identifier test. Equal item identifier merging is an id-based merging criterion and can never imply that the topics represent the same subject of matter. Leaving out the possibility of matching item identifiers, also rules out number 4 and 5 in the equality tests. Point 3, 4 and 5 is omitted if limited merging is enabled. This reduces the number of operations during merging.

If the user has a limited excerpt of a topic map on the mobile phone and she needs other parts of the same topic map, then the id-based merging is appropriate. In such a scenario, the item identifier criteria are the only test needed.

The *mergeByTopicName* criteria is much discussed and there are many strong opinions against the Topic Name Constraint. As mentioned earlier, an equal name in the same scope does not necessarily represent the same subject. To avoid merging between unequal topics with equal names, scopes must be created and added.

When merging topic names, I have not implemented a type check of the topic names. I believe that the usage of a topic name types is not very valuable in the emergency scenario. In the merging of properties the variants are not merged, they are added. I do not believe that the usages of variants are of high importance. A topic can have multiple topic names with different scopes, thus scopes can be used instead. The worst case scenario is redundant variant items in a topic name. The merging of occurrences does not check types or scopes. This is something that could be relevant to implement.

To optimize the algorithm further the associations can be added without any equality tests. This is instead of performing equality tests to detect duplicates. The duplicates are not added.

These opinions and decisions are made out of personal experience, example topic maps and the general understanding of the topic maps concepts. To base the algorithm on empirical data, testing must be performed in an imagined use case scenario.

Adding topic references when merging topics

When merging the properties of two topics it is crucial that the references are maintained. The algorithm for merging topics can be very complex if every reference is checked for equality. If the characteristics of topic B should be added to topic A in topic map *tmA*, the question is how to refer to the correct topic name object. In the example below topic B has a reference to a topic name C that exists in topic map *tmB*. Topic A cannot inherit the reference to a topic in another topic map.

Topic map tmB:

```
<topic id="B">
    <baseName>
           <topicRef xlink: href="#C"/>
    </baseName>
</topic>

<topic id="C">...</topic>
```

Topic map tmA:

```
<topic id="A">...</topic>

<topic id="C2">...</topic>
```

Topic map tmA after merging:

```
<topic id="A">
   <baseName>
           <topicRef xlink: href="#C2"/>
    </baseName>
</topic>

<topic id="C2">...</topic>
```

By using the mergeTable(C, C2), a corresponding topic C2 in tmA is fetched by using topic C as key. Topic C2 is the topic that topic C shall be merged with. The topic A must therefore, be given a reference to topic C2, which is a part of the same topic map. Through the mergeTable, the topics can easily be compared, instead of performing new equality testing. This would be very tedious and time consuming.

Since every item is stored in the filesystem in a location indicated by an element number, this number is used for fetching referenced topics.

6.4.2.4 Merging Associations

Processing associations is an issue to be discussed. The data model does not suggest the merging of association roles, only merging of the reified properties and the item identifiers. I have implemented the method according to the data model. In tinyTIM, associations are added without any form of check. This is also a solution and will be a very good solution if the code is too resource demanding to run on mobile devices. However, if the topic maps used for merging contain very many identical associations, this would lead to a lot of redundancy. In addition, if someone would edit one of the duplicate associations, this would again lead to inconsistency in the merged topic map. When querying for an association, you will not know if you get the edited one or the "old" one. For the emergency scenario, where the intention is not to edit the merged associations, duplicate associations do not necessarily pose a problem. When the application fetches a certain association, it can stop the searching when the first and best one is found. Another aspect is the fact that the ontology is shared and typically, it is here the most general associations would reside, e.g. the relationship between person and worker. The domain specific associations in the *police* topic map will probably not be identical to the associations between the domain specific topics in the *fire* topic map. This is of course speculations and presumptions, but it is fair to say that the omission of this method will most presumably not cause any big problems.

If a topic map engine should implement merging of associations, this will ensure no duplicate associations. The next two examples should preferably be merged into the last example.

Association 1:

```
<association id="parenthood">
    <instanceOf>
      <topicRef xlink: href="#parent-child"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink: href="#father"/>
      </roleSpec>
      <topicRef xlink: href="#frank"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink: href="#child"/>
      </roleSpec>
      <topicRef xlink: href="#david"/>
    </member>
  </association>
```

Association 2:

```
<association id="parenthood">
    <instanceOf>
      <topicRef xlink: href="#parent-child"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink: href="#father"/>
      </roleSpec>
      <topicRef xlink: href="#frank"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink: href="#child"/>
      </roleSpec>
      <topicRef xlink: href="#anna"/>
    </member>
  </association>
```

Merged association:

```
<association id="parenthood">
    <instanceOf>
        <topicRef xlink: href="#parent-child"/>
    </instanceOf>
    <member>
       <roleSpec>
          <topicRef xlink: href="#father"/>
       </roleSpec>
       <topicRef xlink: href="#frank"/>
    </member>
    <member>
      <roleSpec>
         <topicRef xlink: href="#child"/>
      </roleSpec>
      <topicRef xlink: href="#anna"/>
      <topicRef xlink: href="#david"/>
    </member>
</association>
```

When parsing the last XTM, the association roles should be normalized by creating three members for each player of a role, e.g. a member for *frank*, a*nna* and david [40].

6.5 Benchmark Tests

To ensure a smallest possible midlet, I used the ProGuard tool[64]. This tool is a Java class file shrinker, optimiser and obfuscator. It detects and removes unused classes, methods, attributes and fields. It can then optimise byte code by removing unused instructions. At last it renames the remaining classes, methods, attributes and fields with short meaningless names, of the form; a, ab, abc and so on. The obfuscator gives a more compact jar file, which again gives a faster transfer and less storage requirements. These advantages are indeed crucial when developing quite extensive Midlets. When trying to package the MTV without obfuscator the size was 230 KB. When using the obfuscator without any constraints the size

was reduced to 72 KB. Due to dynamic class loading in the code, some of the original class names had to be preserved. The size of the Midlet is 95 KB.

I have developed a midlet to test the merging of topic maps on mobile devices. The midlet has a menu which gives the user the following possibilities,

1. Load topic maps
2. Merge
3. Print merged map, by printing
    a. Topics, or
    b. Associations
4. Delete topic maps

I have only tested the merging of topic maps. Samuel Vigdal tested loading and querying in [6] .To test the system time on mobile devices I have used XTM files of various sizes developed by Samuel Vigdal and myself. The sizes of the topic maps in total were 14 KB, 35 KB and 56 KB for the two topic maps. The documents were developed in Omnigator and they are expressing the emergency domain. The response time will of course vary according to the complexity of the topic map. See appendix A for an example XTM.

I used one mobile device to test the midlet. I chose the fastest mobile that was available. The Sony Ericsson W810 is according to TastePhone[65] ranked as 63 in a mobile benchmark test on the Internet. I have made a table for showing the difference between my test mobile and the strongest mobile according to TastePhone. The differences in processor speed and memory can give a hint of how fast the midlet can run on other more resourceful devices. Both mobiles support the MIDP-2.0 profile and use the CLDC-1.1 configuration.

	RAM (heap size)	**Java virtual processor speed**	**Free RAM at startup**	**Computation performance**	**Free flash memory**	**Performance**
Orange SPV M2000	16415 KB	261.4 MHz	1443 KB	2941	51703 KB	3,878.65
W810	1023 KB	63 MHz	886 KB	420	18347 KB	573.98

The merging algorithm can be performed in a normal or limited version. In addition, the *mergeByTopicName* feature can be enabled or disabled. These settings are used in the benchmark tests and they are tested according to their complexity. The first test uses the limited merging version with *mergeByTopicName* flag set to off. This algorithm makes fewest tests and loops. The most resource demanding test uses the complete merge version and *mergeByTopicName* flag. According to the list below, the complexity and level of detail increases for each test.

1. Limited version and *mergeByTopicName* off
2. Limited version and *mergeByTopicName* on
3. Complete version and *mergeByTopicName* off
4. Complete version and *mergeByTopicName* on

Every test is performed three times to make sure the results are stable. The results had very small deviations.

The average time from every benchmark is inserted into the tables in the next section. The number of topics and associations in the topic map is listed on top of the table. In the last row, the number of topics and associations after the merge is listed. The first topic map is quite small and I had a certain control of which topics had equalities. I made sure that there were several topics and associations that matched in different properties. The results in the last row show the results. In benchmark 2 and 3, the topic maps where much larger and there are fewer equal items.

Benchmark 1 –XTM total size 14 KB
Topic map 1: Before: topics: 21, associations: 2
Topic map 2: Before: topics: 22, associations: 4

1. Limited merge with *mergeByTopicName* off				
	Total merge time	Adding associations	Merging topics	Topics+assoc
Average (secs)	2,84	0,10	0,46	38 + 6

2. Limited merge with mergeByTopicName on				
	Total merge time	Adding associations	Merging topics	Topics+assoc
Average (secs)	3,55	0,09	0,56	26+6

3. Complete merge with mergeByTopicName off				
	Total merge time	Merging associations	Merging topics	Topics+assoc
Average (secs)	4,07	0,12	0,49	32 + 5

4. Complete merge with mergeByTopicName on				
	Total merge time	Merging associations	Merging topics	Topics+assoc
Average (secs)	4,55	0,15	0,55	26+5

The response time increased by approximately 0.50 seconds, for each of the tests. The merging of associations took, in general, ca 20 % -60% more time than the adding of associations, depending on the merging version. The response time for merging topics rested stable during both the limited merge and the complete merge. However, the time increased slightly when the *mergeByTopicName* flag was set (about 10%). Overall, the tests show that the equality tests are the most time-consuming operations. I suggest that simplifications should be applied to this part of the algorithm if the response time is still too long on devices that are more resourceful. As the test showed, the merging of topics was very stable, independent of the merging version. When the *mergeByTopicName* flag was set, the time increased some. The number of topics and associations after the merge had some variations. It was interesting to see that the topic name merge increased the number of merged topics with six. The result showed 26 topics with the *mergeByTopicName* on, and 32 topics without the topic name test. The merging of associations removed one duplicate. The first test resulted in 44 topic map items, while test 4 ended up with 31. This is quite a difference.

Benchmark 2 –total size XTMs : 35 KB
Loading: 3,50 (average, secs)
Topic map 1: Before: topics: 58, associations: 18
Topic map 2: Before: topics: 53, associations: 11

	1. Limited merge with mergeByTopicName off			
	Total merge time	Adding associations	Merging topics	Topics & Assoc
Average (secs)	23,06	0,18	1,1	107+29

	2. Limited merge with mergeByTopicName on			
	Total merge time	Adding associations	Merging topics	Topics & Assoc
Average (secs)	23,60	0,18	1,33	98+29

	3. Complete merge with mergeByTopicName off			
	Total merge time	Merging associations	Merging topics	Topics & Assoc
Average (secs)	29,59	2,85	1,16	99+27

	4. Complete merge with mergeByTopicName on			
	Total merge time	Merging associations	Merging topics	Topics & Assoc
Average (secs)	29,02	2,84	1,42	92+27

When the topic map increased to double size, the merging of associations increased drastically compared to adding associations. The time increased by a factor of almost thirty. The reason why the merging of associations is time-consuming is the fact that they contain many references to other topics. This was also true for the association roles and variants. They where not merged to limit the iterations and decrease system time. As the benchmark displays, the more detailed the algorithm is the more topics are merged. In this topic map, there where obviously no equal associations.

Benchmark 3 –XTM total size 56 KB
Loading: 8, 04 (average, secs)
Topic map 1: Before: topics: 70, associations: 38
Topic map 2: Before: topics: 106, associations: 50

	1. Limited merge with mergeByTopicName off			
	Total merge time	Adding associations	Merging topics	Topics & Assoc
Average (secs)	75,07	0,99	3,79	172+88

	2. Limited merge with mergeByTopicName on			
	Total merge time	Adding associations	Merging topics	Topics & Assoc
Average (secs)	86,78	0,99	4,47	157+88

	3. Complete merge with mergeByTopicName off			
	Total merge time	Merging associations	Merging topics	Topics & Assoc
Average (secs)	136,15	43,89	3,99	157+85

	4. Complete merge with mergeByTopicName on			
	Total merge time	Merging associations	Merging topics	Topics & Assoc
Average (secs)	148,20	42,66	4,61	144+84

According to the last test, the time increased considerably when the associations were merged. This is probably because the topic maps contained almost as many associations as topics. It is possible to skip the merging of associations during the complete merge. However, this could lead to very many identical associations in an already large topic map. The *mergeByTopicName* managed to find 15 topics that matched names and scope. This topic map had very few equal items. You can read from the results that only four topics had PSI or subject locator and fifteen topics had equal topic names. When one topic is equal to another due to the *mergeByTopicName* property is on, this can trigger the equality of other topics as well. If the topic is used as topic type in another topic, this can be decisive in the equality test.

The tests demonstrated that the equality tests are the most resource demanding part of the algorithm. These equality tests iterate *topics_inTMB* * *topics_inTMA*. If topic map tmB has 50 topics and tmA has 50, the algorithm must perform 50*50 tests (250 times). The merging algorithm merges every topic in tmB with a corresponding topic in tmA or an empty one. This is performed 50 times. Even though the merging of topics-algorithm has a more detailed level, the number of loops in the equality tests affects the system time more.

The methods used for performing the tests are qualitative and are based on resources available to me. To gain more detailed and trustworthy test results the number of the tests should be higher and performed on devices that are more resourceful. In addition, different variations within the XTM can also be a good idea. It is impossible to test every possible situation, however The tests will merely be an indicator and a starting point for further scrutiny

7 Discussion, Conclusion and Further Work

The Usage of Topic Maps

The Topic Map standard has many mechanisms that match the requirements in the Ad-Hoc InfoWare project. Topic Maps can express ontologies and support a controlled vocabulary. It is of high importance that the information available is not ambiguous or vague. Topic Maps have many functions that ensure a distinct and explicit expression of information. Typing, scoping and establishing the subject identity are the most important concepts for accomplishing this. The topic map can make it easier to navigate and find the correct information. At a rescue scene, the small details and nuances can be of high importance. The typing mechanism can secure sensitive data by filtering the information items. It can also support the creation of access levels between the organizations. It is crucial that the individual organizations can decide which information is to be shared. The scoping concept can be a very valuable mechanism in the emergency scene. This can add context to the information and it can be possible to query for a certain piece of information. In addition, the usage of associations connects the information together and can give the users a better and complete picture of the situation.

The use of a shared ontology can enable the topic maps to merge across different organizations. This contributes to connecting distributed knowledge and makes it possible to have a global view of the information. The merging procedures have been implemented in the Mobile Topic Viewer and the merging of different sized topic maps proved feasible. The merging process filters out redundant information items. In a resource-weak device, storage is limited and removing unnecessary information is important. It turned out to be too time-consuming to merge associations. This is because they contain many references to other topics. Equality tests create very much iteration. This was also true for the association roles and variants. They where not merged to limit the iterations and decrease system time. In other words, it is probably not possible to ensure a precise result after merging. To keep system time at a minimum, some redundancy must be accepted.

Since the mobile device has limited storage, the user can have a limited excerpt of a topic map on the mobile phone. Whenever she needs another part of the same topic map, the middleware will fetch the necessary parts and remove the superfluous parts. This would justify the use of item identifiers in the equality tests. In this case, the merging process must merge within one topic map. This process should be very easy since a equal item identifiers is the only equality test needed. The shared ontology is a part of every topic map. To avoid redundant associations in the ontology layer, the ontology can be left out when exporting topic maps in the Mobile Ad-hoc Network. The items in the ontology layer can e.g. be typed to separate them from the topic instances. This way the export mechanism gathers the topics that have not been typed as ontoltogy.

Benchmarks

The benchmark tests demonstrated that the most dominating and influential factor was the number of topics and associations that were to be merged. The tests showed that the equality tests are the most resource demanding operations. The limited version omitted the item identifiers in the equality check. The limited version with the *mergeByTopicName* disabled gave the best results in the benchmark. It used 2.84 seconds to merge 43 topics.

Another expensive operation was the merging of associations. The results displayed a time consuming operation, especially in benchmark 2 and 3, where the topic maps contained a lot more associations. I would consider these results as a recommendation to optimise the

implementation or omit the merging of associations. If their merging is omitted, the associations must be added and this could lead to many duplicates. Since the mobile device already has scarce resources, redundant information is not an optimal solution. In a use case where users only merge topic maps from other organizations, redundant items will be a minor problem. Topic maps from different organizations will probably contain different associations. The differences between the complete and the limited merging versions were clear. The larger the topic maps got the more obvious the differences between the versions became. The complete version used more then twice the time compared to the limited version when the topic map size augmented to 55KB.

It is difficult to predict the topic map sizes that will be required at the use case, however I would presume that approximately 100 topics is minimum. The fastest algorithm used 23.06 seconds to merge 111 topics. This is not an acceptable response time. According to the use case requirements, the response time is crucial. However, many mobile phones on the marked inhabit a higher performance. The best-ranked mobile had five times higher performance and approximately four times faster Java Virtual processor compared to the test-mobile. By using better-equipped devices, the system time will definitely improve. There is a constant development within mobile devices. This trend is not likely to stop. Another possibility is to use PDAs for merging topic maps. They are more resourceful and have in addition a larger screen.

Overall, the testing proved that merging on a mobile phone with a medium performance was feasible. The limited version is for the time being the best solution, when looking at the benchmark tests.

Mobile Ad-hoc Network

The usage of mobile devices as a means for communication can e.g. save many repetitive messages between the Emergency Officer and the personnel. In the Mobile Ad-hoc Network, the users can communicate with each other, and messages can be broadcasted. This can be timesaving compared to oral communication. In addition, information can be stored in the topic map and the information can be used later for e.g. analysis.

RDF vs. Topic Maps

Compared to RDF and OWL the elaboration of topic maps can be less verbose since it allows n-ary relationships and has directionless associations. In addition, the algorithms in RDF have been tailored towards reasoning which is a resource demanding mechanism. Topic maps are in comparison a lightweight framework that allows a simpler form of querying and navigating.

Reflections

There is little related work to draw experience from. The standard is, in addition, complex and difficult to grasp. The XTM 1.0 contained shortcomings and the second version is still not approved. Topic Maps have been criticized for being slow in progress in some fields. The lack of good convincing applications can have decelerated the Topic Maps breakthrough. My experience as a newcomer is that the information on Topic Maps is sometimes illegible. There is only one book published on the matter and it is already three years old. The rest of the learning process must go through arbitrary web pages. Topic Maps is a very well suited tool in knowledge management. It increases findability and adds context and semantics to the

information. It has many fine mechanisms and the idea of "Seamless knowledge" is very compelling. However, it seems as if the merging concept has been forgotten. In addition, I believe there are still many unresolved issues concerning a precise and correct merge. Depending on the usage, the algorithm cannot promise to remove every redundancy. The algorithm is complex and can be very deep if every equality test should be performed. The key issue is to "know when to stop". How do you know this? Who decides when to stop? The application developer must decide the constraints of the implementation and solve these issues. It should be elaborated specific guidelines that concern the depth of the merging algorithm.

In the future, the solution should not be limited to the emergency scenario use case. Instead a mobile phone can come in handy in several scenarios; e.g. in the police force, for quick access to information that can be merged with the existing topic map at the police server. It can be used in every aspect possible when information exchange is an issue.

Further work

The algorithm should be further optimised to prune the system time. This can be done by implementing indexes for subject locators, item identifiers and PSIs. These items are used in the equality tests, and are therefore a part of the bottleneck. If this is accomplished it is possible to consider the merging of association roles. This can contribute to the removal of redundant items

1. Improving the merging
    a. Implement further indexes
    b. merge association roles
    c. merge variants
    d. find ways to optimise the equality tests
    e. Making it possible to delete desired parts of the topic map to save storage

2. Improving MTV
    a. Improve the removal of topics and topic maps
    b. Convert to data model 1.0
    c. Implement reification
    d. Adopt to the XTM 2.0

3. Testing
    a. Test merging within one topic map
    b. Test serialization after merging

The next step is to test on more resourceful mobile devices and test with larger topic maps.

In addition there is a need for a more thorough mapping of the emergency use case scenario. The needs of the emergency personnel can be found by performing interviews and perhaps by attendance at an emergency rehearsal. More information on these issues can contribute to a better understanding of the way the topic map system can be used.

8 References

1. Plagemann, T., et al., *Middleware Services for Information Sharing in Mobile Ad-hoc Networks - Challenges and Approach.*, in *Workshop on Challenges of Mobility, IFIP TC6 World Computer Congress, 2004,*, D.M. Systems, Editor. 2004: Toulouse, France. p. project description.

2. Wikipedia. *Ontology*. 2006 [cited 2006 02.07.2006]; An online encyclopedia]. Available from: http://en.wikipedia.org/wiki/Ontology.

3. W3C. *Resource Description Framework (RDF)*. 2002 19.06.2006 [cited 2006 07.03]; 1.176:[Specification document for RDF]. Available from: http://www.w3.org/RDF/.

4. ISO/IEC. *ISO/IEC 13250 Topic Maps*. 2002 08.07 [cited 2006 02.02]; Available from: http://www1.y12.doe.gov/capabilities/sgml/sc34/document/0322.htm.

5. Schwotzer, T. and K. Geis, *Shark-a System for Management, Synchronization and Exchange of Knowledge in Mobile User Groups.* Journal of Universal Computer Science, 2002. **8**(6): p. 644-651.

6. Vigdal, S., *Informasjonsdeling i redningsarbeid ved bruk av Emnekart (Topic Maps)*, in *Departement of Informatics*. 2006, University of Oslo: Oslo. p. 68.

7. Plageman, T. *Ad-Hoc InfoWare, Middleware Services for Information Sharing in Ad-hoc Networks*. [www] 2003 2005 [cited 2006 01.02]; project description]. Available from: http://www.ifi.uio.no/~infoware/.

8. Goebel, V. and Ellen Munthe-Kaas. *Employing Ontologies for Information Sharing in Mobile Ad-hoc Networks*. 2005 [cited 2006 10.10]; Master thesis description]. Available from: http://www.ifi.uio.no/dmms/course.php?course_id=102.

9. Sikkerhetsdagene. *Sikkerhetsdagene*. 2006 11.7.2006 [cited 2006 15.07]; Safety conference in Trondheim]. Available from: http://www.sikkerhetsdagene.no/.

10. Direktoratet for Samfunnssikkerhet og beredskap. *Mer effektiv redningsinnsats*. 2006 16.02.2006 [cited 2006 03.03]; A description of NARRE]. Available from: http://dsb.no/Searchadv.asp?search=true.

11. Direktoratet for Samfunnssikkerhet og beredskap. *Krisehåndtering*. 2002 [cited 2006 04.05.2006]; Available from: http://dsb.no/article.asp?ArticleID=1033&Rank=1&SubRank=3.

12. Justis-og politidepartementet. *Den Norske Redningstjeneste*. 2001 [cited; A description of the Norwegian rescue service].

13. Helse ØST. *Regional plan for helsemessig og sosial beredskap i helse Øst*. 2005 June 2005 [cited 2006 05.05]; 2.0:[Available from: http://www.helse-ost.no/modules/module_123/proxy.asp?D=2&C=221&I=767&mids=166a302.

14. The TETRA MoU Association Ltd. *TETRA*. 2006 [cited 2006 02.10]; Available from: http://www.tetramou.com/.

15. Pepper, S. *Euler, Topic Maps and Revolution*. 1999 [cited 2006 05.10]; Available from: http://www.infoloom.com/tmsample/pep4.htm.

16. Biezunski, M., S. Newcomb, and M Bryan. *Guide to the topic map standards*. Information Technology Document Description and Processing Languages 2006 [cited 2006 02.03.2006]; Available from: http://www1.y12.doe.gov/capabilities/sgml/sc34/document/0323.htm.

17. W3C. *The Semantic Web*. 2006 [cited 2006 01.07.2006]; Available from: http://www.w3.org/2001/sw/.

18. Park, J. and S. Hunting, *XML Topic Maps creating and using Topic Maps for the web*. 1st ed, ed. J. Park. 2003, Boston: Addison-Wesley Professional. 544.

19. Waggoner, B. *Carl Linnaeus*. 1996 01.01.2000 [cited 2006; Available from: http://www.ucmp.berkeley.edu/history/linnaeus.html.

20. Mindswap.org. *Debugging OWL Ontologies using Swoop*. 2006 [cited 2006 08.06.2006]; Ongoing research]. Available from: http://www.mindswap.org/2005/debugging/.

21. Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. [cited 2006 05.06.2006]; Introduction to ontology]. Available from: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

22. Tanasescu, V., et al. *A Semantic Web Services GIS based Emergency Management Application*. 2006 [cited 2006 02.07]; Paper on a project description]. Available from: http://kmi.open.ac.uk/projects/dip/resources/iswc06/SemanticWebChallenge2006_DIP.pdf.

23. Stefanelli, M., G. Leonardi, and S. Panzarsa. *Web-based Health Service Flow Management System*. 2005 [cited 2006 02.10]; Available from: http://www.labmedinfo.org/research/serviceflow/index.htm.

24. W3C. *OWL Web Ontology Language*. 2004 10.02.2004 [cited 2005 12.08]; Overview]. Available from: http://www.w3.org/TR/owl-features/.

25. Garshol, L.M. *Practical Topic Maps using the OKS*. 2005 07.02 [cited; 1.0:[Course material from OKS seminar].

26. Garshol, L.M., *Practical Topic Maps using the OKS*. 2005, Ontopia AS.

27. Garshol, L.M. and S. Pepper, *Only Connect*.... 2005, Ontopia as.

28. Garshol, L.M., *Metadata? Thesauri? Taxonomies? Topic Maps! Making sense of it all.* Journal of Information Science, 2004. **30**(4): p. 378-391.

29. TopicMaps.Org Authoring Group. *XML Topic Maps (XTM) 1.0*. 2001 29.10.2002 [cited 2006 02.04.2006]; version 1.16:[Available from: http://topicmaps.org/xtm/.

30. Garshol, L.M., *Practical Topic Maps using the OKS*. 2006, Ontopia AS.

31. Garshol, L.M., *The Linear Topic Map Notation*, L.M. Garshol, Editor. 2006, Ontopia AS: Oslo.

32. Barta, R. and L. Heuer. *AsTMa= 2.0 Language Definition*. 2005 2005-11-24 [cited 2006 01.07.2006]; 2.0:[Technical report]. Available from: http://astma.it.bond.edu.au/astma=-spec-2.0r1.0.dbk.

33. JTC1/SC34/WG3, I.I. *Topic Map Query Language (TMQL)*. 2005 2005 [cited 2006 05.06.2006]; Document the efforts to create the ISO 18048 standard as part of the Topic Map standards family.]. Available from: http://www.isotopicmaps.org/tmql/.

34. Ontopia. *The Ontopia Schema Language*. 2005 12.1.2006 [cited 2006 02.07]; version 3.0:[Available from: http://www.ontopia.net/omnigator/docs/schema/spec.html.

35. Garshol, L.M. and Graham Moore. *Topic Maps — Data Model*. ISO/IEC JTC1/SC34 2006 18.06.2006 [cited 2006 09.06.2006]; Final Draft International Standard]. Available from: http://www.isotopicmaps.org/sam/sam-model/.

36. Pepper, S. *The TAO of Topic Maps*. 2000 2002 [cited 2006 02.03]; An introduction to topic maps]. Available from: http://www.ontopia.net/topicmaps/materials/tao.html.

37. OASIS. *OASIS*. 2006 [cited 2006 02.08]; Home page]. Available from: http://www.oasis-open.org/home/index.php.

38. Garshol, L.M., G. Moore, and J. SC34. *Topic Maps — XML 2.0*. 2006 19.06.2006 [cited 2006 01.08]; 2.0:[Final draft for the XML 2.0]. Available from: http://www.isotopicmaps.org/sam/sam-xtm/2006-06-19/.

39. Pepper, S. and G.O. Grønmo, *Towards a General Theory of Scope*, Ontopia, Editor. 2002, Ontopia.net: Oslo. p. Paper on the scope issue.

40. Moore, G., Kal Ahmed, and Lars Marius Garshol. *The Topic Map API*. 2005 05.04.2006 [cited 2006 02.04]; Available from: http://tmapi.org/apiDocs/index.html.

41. Heuer, L. *tinyTIM - a tiny Topic Map Engine & TMAPI implementation*. 2005 2005 [cited 2006 01.04]; Homepage and link to download the tinyTIM topic map engine]. Available from: http://tinytim.sourceforge.net/.

42. Ontopia. *The Omnigator*. 2005 [cited 2006 01.02]; Free showcase]. Available from: http://www.ontopia.net/omnigator/models/index.jsp.

43. Ontopia. *Getting Started with Topic Maps*. 2005 [cited 2005 02.05]; Introduction to Ontopia's tools]. Available from: http://www.ontopia.net/topicmaps/index.html.

44. Ahmed, K. and C. Frohlich. *TM4J*. 2006 2004 [cited 2006 02.10]; Project page for TM4J]. Available from: http://tm4j.org/.

45. Ahmed, K. *TM4J*. 2005 [cited 2006 02.07]; Project description]. Available from: http://tm4j.org/about.html.

46. SourceForge.com. *Perl XTM*. 2006 [cited 2006 02.05]; Project download]. Available from: http://sourceforge.net/projects/perlxtm/.

47. NetworkedPlanet. *TMCore*. 2005 [cited 2006 02.03]; Homepage for a commercial engine]. Available from: http://networkedplanet.com/.

48. Kongsbakk, I., *BIBSYS Bruk av emnekart i biblioteksystem*. 2005, University of Oslo. p. 30.

49. Gulbrandsen, A., *Houston Emnekart - Utdrag av prosjektrapport*. 2003, University of Oslo: Oslo. p. 20.

50. Woodman, M. *Topic Maps tools*. 2006 [cited 2006 02.10]; Available from: http://www.topicmap.com/topicmap/tools.html.

51. Bray, T. and Joshua Tauberer. *What is RDF?* 1998 2006 [cited 2006 13.09]; 3:[Available from: http://www.xml.com/pub/a/2001/01/24/rdf.html?page=2.

52. Garshol, L.M. *Living with topic maps and RDF*. 2004 [cited 2005 10.10]; Available from: http://www.ontopia.net/topicmaps/materials/tmrdf.html#N106.

53. Ontopia. *The RTM RDF to topic maps mapping Definition and introduction*. 2003 [cited 2006 02.03]; 0.2:[This technical report defines version 0.2 of the RTM RDF to topic maps mapping vocabulary.]. Available from: http://www.ontopia.net/topicmaps/materials/rdf2tm.html.

54. ISO/IEC. *ISO/IEC 13250 Topic Maps*. 1999 08.07 [cited 2006 02.02].

55. Seedorf, S., A. Korthaus, and M. Aleksy. *Creating a topic map query tool for mobile devices using J2ME and XML*. in *WISICT '05*. 2005. Cape Town, South Africa: Trinity College Dublin.

56. Maicher, L. and Hans Friedrich Witschel. *Merging of Distributed Topic Maps on the Subject Identity Measure (SIM) Approach*. in *Proceedings of Leipziger Informatiktage (LIT)*. 2004. Leipzig.

57. Wikipedia. *Knowledge base*. 2006 17.09.2006 [cited 2006 17.09]; Available from: http://en.wikipedia.org/wiki/Knowledge_base.

58. Sun Microsystems, I. *Java ME At a Glance*. 2006 [cited 2006 17.05.2006]; Available from: http://java.sun.com/javame/index.jsp.

59. Schwotzer, T. *Shark - Mobile Shared Knowledge 2005* [cited 2006 05.07]; project description]. Available from: http://kbs.cs.tu-berlin.de/ivs/Projekte/Shark/index_en.html.

60. Ahmed, K. *The TMShare Application*. 2003 [cited 2006 05.10]; Available from: http://www.techquila.com/topicmapster_2.html.

61. Ontopia. *Vizigator*. 2006 [cited 2006 02.10]; Information on the Vizigator and download]. Available from: http://www.ontopia.net/solutions/vizigator.html.

62. Pepper, S. and L.M. Garshol, *Seamless Knowledge with TMRAP*, in *Extreme Markup Conference 2004*. 2004, Ontopia: Montréal, Quebec.

63. Sun Microsystems, I. *Java(tm) 2 Platform Standard Edition 5.0 API Specification*. 2005 [cited 2006 01.02.2006]; 5.0:[The Java API]. Available from: http://java.sun.com/j2se/1.5.0/docs/api/overview-summary.html.

64. SourceForge.com and E. Lafortune. *ProGuard*. 2005 [cited 2006 02.07]; 3.6:[Available from: http://proguard.sourceforge.net/.

65. TastePhone. *TastePhone*. 2006 [cited 2006 05.10]; MIDP telephones benchmark]. Available from: http://www.club-java.com/TastePhone/J2ME/MIDP_Benchmark.jsp;jsessionid=DED64274E52066FF281A6038C665BA95.

Appendix A Methods Overview

Methods that are added or changed in the Mobile Topic Viewer:

MTopic.java:

- getRolesPlayed()
- setRolesPlayed(AssociationRole assocRole)
- mergeIn(Topic other)
- mergeIn(Topic other, boolean modify, Hashtable mergeMap)
- checkScope(Set thisScoped, Set otherScoped, Hashtable lookup)

MTopicMap.java:

- mergeIn(TopicMap otherTopicMap)
- mergeAssociations(TopicMap otherTopicMap, Hashtable mergeTable)
- addAssociations(TopicMap otherTopicMap, Hashtable mergeTable)
- getAssociationNodes(Set oSet_, INode oNode_)
- checkScope(Set thisScoped, Set otherScoped)
- setMergeByTopicNameFlag

Properties.java

- added a property for setting the merge to limited or complete, limited is set to true by default

XTMParser.java:

- parseMember (Association oAssociation_) , added code for setting the associationRoles to the appropriate topic when parsing the XTM.

MAssociationsIndex.java:

- GetAssociationTypes ()
- GetAssociationByType (Topic topic)

ISConst.java:

- Added the static variables for the XTM I wanted to use.

The Midlet:
Merge.java:

- printMergedMap ()
- loading()
- merge()
- deleteTopicMaps()
- getTopicNamesString(Topic t)

Appendix B Algorithms

The most important methods in MTopicMap.java:

```
public boolean getMergeByTopicNameFlag(){
	boolean mergeByTopicName = false;
	try {
		TopicMap tmTmp =this.getTopicMap();
		TopicMapSystem tmSystem = tmTmp.getTopicMapSystem();
			mergeByTopicName =
			tmSystem.getFeature("http://tmapi.org/features/merge/byTopicName");

	} catch (FeatureNotRecognizedException e) {
			e.printStackTrace();
	}
	return mergeByTopicName;
}
public boolean getLimitedFlag(){
	boolean limited = false;
	String flag = this.getTopicMapSystem()
	.getProperty(ISConst.STR_MERGE_LIMITED);
	if(flag.equals("1")){
		limited = true;
		oLogger.info("Limited merging on");
	}else{
		oLogger.info("Complete merging on");
	}
	return limited;
}

public void mergeIn(TopicMap otherTopicMap) {

/* the table for storing the mergeable topics */
Hashtable mergeTable = new Hashtable();

/* the table for storing the non-mergeable topics */
Collection createList = new MCollection();

boolean mergeByTopicName =getMergeByTopicNameFlag();
boolean limited =getLimitedFlag();

boolean merge;
Iterator it1 = otherTopicMap.getTopics().iterator();
	while(it1.hasNext()){
		MTopic otherTopic = (MTopic)it1.next();
		merge = false;
		MTopic thisTopic = null;
		boolean diffSubject = false;
		Iterator it2 =getTopics().iterator();
		while((merge == false) && (it2.hasNext())){
		//1. test identical subject locators "the topic"
			thisTopic = (MTopic)it2.next();
			Set otherSL = otherTopic.getSubjectLocators();
			Set thisSL = thisTopic.getSubjectLocators();
			if((otherSL.size()) > 0 && (thisSL.size()) > 0){
					Iterator it = otherSL.iterator();
				while(it.hasNext()&& merge == false){
				  Locator loctest = (Locator)it.next();
				  Iterator itTmp = thisSL.iterator();
				 while(itTmp.hasNext()){
```

```
                                    Locator loc2 = (Locator)itTmp.next();

                                    if( loctest.equals(loc2)) {
                                        merge = true;

                                    }
                                    else{
                        /* the subjects are not equal, no need to continue testing */
                                    diffSubject = true;
                                    }
                                }
                            }
                        }

                        //2. test identical subject identifier "PSI"
                        if(merge == false && diffSubject == false){
                                Set otherSI = otherTopic.getSubjectIdentifiers();
                                    Set thisSI = thisTopic.getSubjectIdentifiers();
                                    if((otherSI.size()) > 0 && (thisSI.size()) > 0){
                                            Iterator it3 = otherSI.iterator();
                                        while(it3.hasNext()&& merge ==false){
                                            Locator loc3 = (Locator)it3.next();
                                            Iterator it4 = thisSI.iterator();
                                            while(it4.hasNext()){
                                                Locator tmp = (Locator)it4.next
                                                 if( loc3.equals(tmp)){
                                                        merge = true;
                                                }
                                                else{
                        /* the subjects are not equal, no need to continue testing */
                                                        diffSubject = true;
                                                }
                                            }
                                        }
                                    }
                        }
//                      3. test identical source locator/item identifier "xtm#id"
                          if(merge == false && limited == false && diffSubject == false){
                            Iterator it4 = otherTopic.getSourceLocators().iterator();
                            while(it4.hasNext()&& merge ==false){
                                    Locator loc = (Locator)it4.next();
                                    if(thisTopic.getSourceLocators().contains(loc))
                                            merge = true;
                                    }
                            }
                       }
                    //4. test if the PSI is identical to the source loc/item identifier
                    if(merge == false && limited == false && diffSubject == false){
                            Iterator it5 = otherTopic.getSubjectIdentifiers().iterator();
                            while(it5.hasNext()&& merge ==false){
                                    Locator loc = (Locator)it5.next();
                                 if(thisTopic.getSourceLocators().contains(loc)){
                                            merge = true;
                                    }
                            }
                    }
                    //5. test the opposite
                    if(merge == false && limited == false && diffSubject == false){
                            Iterator it6 = otherTopic.getSourceLocators().iterator();
                            while(it6.hasNext()&& merge ==false){
                                    Locator loc = (Locator)it6.next();
                                if(thisTopic.getSubjectIdentifiers().contains(loc)){
                                            merge = true;
```

```
                                        }

                                }
                        }
                        //if feature :mergebytopicname, is supported:
                        //6. the names must be equal and in the same scope */
                        if(mergeByTopicName){
                                if(merge == false && diffSubject == false){
                                        Iterator it7 = otherTopic.getTopicNames().iterator();

                                        while(it7.hasNext()&& merge ==false){
                                            TopicName nme = null;

                                             nme = (TopicName)it7.next();
                                            Set otherScoped = nme.getScope();
                                            Iterator it10 = thisTopic.getTopicNames().iterator();
                                            //prevents uneccessary loops
                                           while(it10.hasNext() && merge == false){
                                                TopicName name = (TopicName)it10.next(
                                                Set thisScoped = null;
                                                if(name.getValue().equals(nme.getValue())){

                                                     thisScoped = name.getScope();
                                                 merge =checkScope(thisScoped,otherScoped);


                                                }
                                            }
                                        }
                                }

                        }//end mergeByTopicName
                }//end looping the topics from 'this' topic map
                if( merge){
                        mergeTable.put(otherTopic, thisTopic);                                                  }
                else{

                        boolean object = createList.add(otherTopic);
                        if(!object){
                                throw new NullPointerException("The object added in " +
                                                                "createList is null"+otherTopic.getObjectId());
                        }
                }
        }//end outer loop,topics from other are all checked :)

        //creating a new topic in this topic map->to be merged with the topic from 'other'
        Iterator it7 = createList.iterator();
        while(it7.hasNext()){
                MTopic topic = (MTopic)it7.next();
                MTopic emptyTopic = (MTopic)createTopic();
                mergeTable.put(topic, emptyTopic);
        }
        // merging the  topics in the mergetable
        long startTime = System.currentTimeMillis();
        for (Enumeration num = mergeTable.keys() ; num.hasMoreElements() ;){
                MTopic othTopic = (MTopic)num.nextElement();
                MTopic thistopic = (MTopic)mergeTable.get(othTopic);
                thistopic.mergeIn(othTopic,false,mergeTable);
        }
        long stopTime = System.currentTimeMillis();
        long runTime = (stopTime - startTime);

        oLogger.info("Time merging topics: "+Long.toString(runTime));
```

```
// add associations, if duplicates add source locator and scope
if (!limited){
        long startTime2 = System.currentTimeMillis();
        mergeAssociations( otherTopicMap,  mergeTable);
        long stopTime2 = System.currentTimeMillis();
        long runTime2 = (stopTime2 - startTime2);
        oLogger.info("Time merging assocs: "+Long.toString(runTime2));
}
else{
        long startTime2 = System.currentTimeMillis();
        addAssociations(otherTopicMap,  mergeTable);
        long stopTime2 = System.currentTimeMillis();
        long runTime2 = (stopTime2 - startTime2);

        oLogger.info("Time adding assocs: "+Long.toString(runTime2));
}

}//end method

private void addAssociations(TopicMap otherTopicMap, Hashtable mergeTable){
        Iterator it =otherTopicMap.getAssociations().iterator();
        while(it.hasNext()){
                Association newOne = createAssociation();
                Association assocOther = (Association)it.next();
                Iterator it10 = assocOther.getAssociationRoles().iterator();
                while(it10.hasNext()){
                        AssociationRole aRole =(AssociationRole)it10.next();
                        Topic newRole = (Topic)mergeTable.get(aRole.getType());
                        Topic newPlayer = (Topic)mergeTable.get(aRole.getPlayer())

                        if(newPlayer == null){
                                throw new TMAPIRuntimeException("There is an association with players
which is not" +
                                                                "in the TopicMap");
                        }
                        if(newRole == null){
                            throw new TMAPIRuntimeException("There is an association with roles which is
not" +
                                                                "in the TopicMap");
                        }
                        newOne.createAssociationRole(newPlayer, newRole);
                        Locator sl = (Locator)assocOther.getSourceLocators().iterator().next();
                        if(sl != null){
                                newOne.addSourceLocator(sl);
                        }
                        if(assocOther.getType() != null){
                                newOne.setType(assocOther.getType());
                        }
                }
        }
}

private void mergeAssociations(TopicMap otherTopicMap, Hashtable mergeTable){
        Iterator it9 =otherTopicMap.getAssociations().iterator();
        while(it9.hasNext()){
                boolean duplicat= false;
                boolean copy = true;
                Iterator assocIt = getAssociations().iterator();
                Association assocOther = (Association)it9.next();
                Association assocThis = null;
                // merge if [type], [scope] and [assocroles] are equal
                while(assocIt.hasNext() && copy){
                        assocThis = (Association)assocIt.next();
```

```
                Topic typeThis = (Topic)assocThis.getType();
               Topic typeOther = (Topic)mergeTable.get(assocOther.getType());

                //[type]
                if (typeThis == null && typeOther == null){
                        duplicat = true;
                }
                else if(typeThis.equals(typeOther)) {
                        duplicat = true;
                }
                // [associationsroles]TODO check identical [scopes]
                if(duplicat){
                        Iterator itRoles = assocOther.getAssociationRoles().iterator();
                        int size =assocOther.getAssociationRoles().size();
                        int count =0;
                        while(itRoles.hasNext()){
                                duplicat = false;
                                AssociationRole assocRole = (AssociationRole)itRoles.next();
                                //gets the type of role e.g. writer

                                Topic roleOther = (Topic)mergeTable.get(assocRole.getType());

                                Topic player = (Topic)mergeTable.get(assocRole.getPlayer());
                                Iterator itRolesThis = assocThis.getAssociationRoles().iterator();
                                while(itRolesThis.hasNext()){
                                        AssociationRole assocRoleThis =
                                        (AssociationRole)itRolesThis.next();
                                        Topic roleThis = (Topic)assocRoleThis.getType();
                                        Topic playerThis = (Topic)assocRoleThis.getPlayer();

                                        if(roleThis.equals(roleOther) &&
                                        playerThis.equals(player)){
                                        duplicat = true;
                                                count++;

                                        }

                                }
                        }

                        if(size == count){
                                copy = false;
                        }else{
                                duplicat = false;
                        }
                }

        }//end inner while loop
        Locator sl = (Locator)assocOther.getSourceLocators().iterator().next();
        if(copy){
                Association newOne = createAssociation();
                Iterator it10 = assocOther.getAssociationRoles().iterator();
                while(it10.hasNext()){
                        AssociationRole aRole =(AssociationRole)it10.next();
                        //gets the type of role e.g. writer
                        Topic newRole = (Topic)mergeTable.get(aRole.getType());
                        Topic newPlayer = (Topic)mergeTable.get(aRole.getPlayer());

                        if(newPlayer == null){
                                        throw new TMAPIRuntimeException("There is an
                                association with players which is not" +
                                                        "in the TopicMap");
                        }
```

```
                                if(newRole == null){
                                        throw new TMAPIRuntimeException("There is an association with
                                roles which is not" +
                                                                        "in the TopicMap");
                                }
                                newOne.createAssociationRole(newPlayer, newRole);

                        }

                        Topic types =(Topic)mergeTable.get(assocOther.getType());
                        if(types != null){

                                newOne.setType((Topic)mergeTable.get(assocOther.getType()));

                          }
                        if(sl != null){
                                newOne.addSourceLocator(sl);
                        }
                }
                else{
                        if(sl != null){
                                assocThis.addSourceLocator(sl);
                        }
                }
        }
}
//cannot perform a lookup in mergeTable to find correct reference since the table is not //filled yet
private boolean checkScope(Set thisScoped, Set otherScoped){
        boolean merge = false;
        if(otherScoped.isEmpty() && thisScoped.isEmpty()){
                merge = true;

        }
//          apply a limited equality-check on the scope, checks only ONE scope
        else if (!otherScoped.isEmpty() && !thisScoped.isEmpty()){

                Topic otherScope = (Topic)otherScoped.iterator().next();
                Topic thisScope = (Topic)thisScoped.iterator().next();
                if(!(otherScope.getSourceLocators()).isEmpty() && !(
                        thisScope.getSourceLocators().isEmpty())){

                        if((otherScope.getSourceLocators()
                                .iterator().next()).equals(
                                thisScope.getSourceLocators()
                                .iterator().next())){
                                merge = true;
                        }
                }
                if(!(otherScope.getTopicNames().isEmpty()) && !(thisScope.getTopicNames().isEmpty())){
                        if((otherScope.getTopicNames()
                        .iterator().next()).equals(
                        thisScope.getTopicNames().iterator().next())){
                                merge = true;
                        }
                }
                else if(!(otherScope.getSubjectIdentifiers().isEmpty()) &&
                        !(thisScope.getSubjectIdentifiers().isEmpty())){

                        if(otherScope.getSubjectIdentifiers()
                        .iterator().next().equals(

                        thisScope.getSubjectIdentifiers()
                                .iterator().next())){
```

```
                                        merge = true;
                                }
                        }
                }
                return merge;
}
```

The most important methods from MTopic.java:

```
public void mergeIn(Topic other,boolean modify, Hashtable mergeTable) throws MergeException
{
        boolean mergeByTopicName = getMergeByTopicNameFlag();                   //merge  subject
locator,it is an error if they have different SLs                       Set                     otherSLs
=other.getSubjectLocators();
        if(!otherSLs.isEmpty()){
                Locator otherLocator = (Locator)otherSLs.iterator().next();
                Locator thisLoc =(Locator)this.getSubjectLocators().iterator().next();
                        if(thisLoc != null && otherLocator != null){
                if(!thisLoc.getReference().equals(
                                        otherLocator.getReference())){
                                        throw new SubjectLocatorClashException(this, other, "the topics
                                        have " +"different subject locator.");
                                }
                        }
                        else{
                                this.addSubjectLocator(otherLocator);
                        }
        }
        //merge [roles played]
        Iterator otherRoles = other.getRolesPlayed().iterator();
        while(otherRoles.hasNext()){
                AssociationRoleassocRole= (AssociationRole)otherRoles.next();

                Topic player =(Topic) mergeTable.get(assocRole.getPlayer());
                Topic role =(Topic) mergeTable.get(assocRole.getType());
                //TODOcheck equality, theother assocRole of theparent
                // association  must be checked
                assocRole.setPlayer(player);
                assocRole.setType(role);
                this.setRolesPlayed(assocRole);

        }
        //merge [types]
        Iterator it =other.getTypes().iterator();
        while(it.hasNext()){
                boolean insertType = true;
                Topic type =(Topic)it.next();

                Iterator it2 =this.getTypes().iterator();
                Set thisTypes = this.getTypes();
                if(!thisTypes.isEmpty()){
                        while(it2.hasNext()&& insertType){
                                Topic thisType =(Topic)it2.next();
                                Topic lookUp = (Topic)mergeTable.get(type);
                                if(lookUp.equals(thisType)){
                                        insertType = false;

                                }
                        }

                }
```

```
                if(insertType){ //gets the correct reference
                        Topic type2 =(Topic) mergeTable.get(type);
                        this.addType(type2);
                }
        }
        //merge source locators/item identifiers
        Iterator it5 = other.getSourceLocators().iterator();
        while(it5.hasNext()){
                boolean insert = true;
                Locator sl = (Locator)it5.next();
                Iterator thisLocators = this.getSourceLocators().iterator();
                while(thisLocators.hasNext()&& insert){
                        Locator thisSource = (Locator)thisLocators.next();
                        if(thisSource.getReference().equals(sl.getReference())){
                                insert = false;
                        }
                }
                if(insert){
                        this.addSourceLocator(sl);
                }
        }
        //merge PSIs
        Iterator it2 = other.getSubjectIdentifiers().iterator();
        while(it2.hasNext()){
                Locator si = (Locator)it2.next();
                Set tmpSet =this.getSubjectIdentifiers();
                if(tmpSet.isEmpty()){
                        this.addSubjectIdentifier(si);
                }
                else{
                        Iterator it8 =this.getSubjectIdentifiers().iterator();
                        while (it8.hasNext()){
                                Locator si2 = (Locator)it8.next();
                                if(!(si.equals(si2))){
                                        this.addSubjectIdentifier(si);

                                }
                        }
                }
        }
        /* Merge topic names: two topic names are considered equal if they have:
         * identical [value] (the string name)
         * TODO identical [type] (seldom case)
         * identical [scope]
         * TODO: merge variant names
         */
        Iterator it3 = other.getTopicNames().iterator();
        boolean merge;
        while(it3.hasNext()){
                merge = false;
                TopicName otherName = (TopicName)it3.next(); //[value]                                Set
otherScoped = otherName.getScope(); //[scope]
                Topic otherType = (Topic)lookup .get (otherName.getType());
                Set variantSet =otherName.getVariants();
                if(!mergeByTopicName){
                        addTopicName(otherName,otherScoped,otherType,variantSet);

                }
                else{
                 Set testSet = this.getTopicNames();
                        if(testSet.isEmpty()){       //add properties                        addTopicName
                                (otherName,otherScoped,otherType,variantSet);
```

```
                        }
                        else{
                                Iterator it4 = this.getTopicNames().iterator();
                                TopicName thisName = null;
                                while(it4.hasNext()&& !merge){
                                        thisName = (TopicName)it4.next();
                                        Set thisScoped =thisName.getScope();
                                        if(thisName.getValue()
                          .equals(otherName.getValue())){

                                                merge =checkScope
                                  (thisScoped,otherScoped,lookup);
                                        }
                                }
                                if(!merge){
                                addTopicName
                                        (otherName,otherScoped,otherType,variantSet);
                                }
                                else{//add source locators and variants

                                    Iterator iter =otherName        .getSourceLocators().iterator();
                                   while(iter.hasNext()){
                                                Locator loc =(Locator) iter.next();
                                                        thisName.addSourceLocator(loc )
                                        }
        //                                TODO : merge variant names
                                        if(!variantSet.isEmpty()){

                                             Iterator varIt = variantSet.iterator();
                                              while(varIt.hasNext()){

                                                Variant var = (Variant)varIt.next();
                                                if(var.getResource()!= null){
                                                 thisName.createVariant
                                (var.getResource(),var.getScope());

                                                }
                                                else{

                                    thisName.createVariant(
                                                  var.getValue(),var.getScope());
                                                }
                                        }
                                }
                        }
                }
        }
}
//Merge occurrences: [scope], [type], [value] and [datatype] must be equal
Iterator it4 = other.getOccurrences().iterator();
while(it4.hasNext()){
        Occurrence occurr = (Occurrence)it4.next();
        merge = false;
        Iterator it6 = this.getOccurrences().iterator();
        Occurrence occurr2 = null;
        while(it6.hasNext()){
                occurr2 = (Occurrence)it6.next();
                // checking [value] and [datatype] (inline characters)
                if((occurr2.getValue() !=null) && (occurr.getValue() !=null)){
                        if(occurr2.getValue().equals(occurr.getValue())){
                                //TODO check [type] and [scope]
                                merge = true;
                        }
```

```
                }

                //checking [value] and [datatype] (external resource)
                elseif((occurr2.getResource())!= null && (occurr.getResource() !=null)){
                        if(occurr2.equals(occurr) ){
        //                              TODO check [type] and [scope]
                                merge = true;
                        }
                }
        }
        if(merge){// add source locators
                Iterator iter =occurr.getSourceLocators().iterator();
                while(iter.hasNext()){
                        Locator loc = (Locator)iter.next();
                        occurr2.addSourceLocator(loc);
                }
        }
        else{// create new occurrence
        //Topic type = (Topic)occurr.getType();
        if(occurr.getValue() !=null){
        createOccurrence(occurr.getValue(),null,occurr.getScope());

        }
        else if(occurr.getResource() !=null){
          createOccurrence(occurr.getResource(),null,occurr.getScope());

        }
   }
}
// TODO if merging is within the same topic map, the other topic must be removed!
// the remove method writes to the persistent storage...
if(modify){
        try{
                other.remove();
        }
        catch(TopicInUseException e){
                throw new TMAPIRuntimeException(e);
        }
}
}

/**
 * This is used when merging topics internally
*/
public void mergeIn(Topic other) throws MergeException
{
        mergeIn(other, true, null);
}

private boolean checkScope(Set thisScoped, Set otherScoped, Hashtable lookup){
        boolean merge = false;
        if(otherScoped.isEmpty() && thisScoped.isEmpty()){
                merge = true;
        }
         //checks only one scope
        else if (!otherScoped.isEmpty() && !thisScoped.isEmpty()){
                Topic otherScope = (Topic)otherScoped.iterator().next();
                Topic thisScope = (Topic)thisScoped.iterator().next();
                if(!(otherScope.getSourceLocators()).isEmpty() && !(
                        thisScope.getSourceLocators().isEmpty())){
                        if((otherScope.getSourceLocators() .iterator().next()).equals(
                          thisScope.getSourceLocators().iterator().next())){
                                        merge = true;
```

```
                    }
                }
                if(!(otherScope.getTopicNames().isEmpty())&& !(thisScope.getTopicNames().isEmpty())){

                    if((otherScope.getTopicNames().iterator() .next()).equals(
                                thisScope.getTopicNames().iterator().next())
                        merge = true;
                    }
                }
                else if(!(otherScope.getSubjectIdentifiers().isEmpty()) &&
                    !(thisScope.getSubjectIdentifiers().isEmpty())){

                if(otherScope.getSubjectIdentifiers().iterator()
                            .next().equals(
                            thisScope.getSubjectIdentifiers().iterator().next())){

                            merge = true;
                    }
                }
                if(thisScope.equals(otherScope)){
                    merge = true;
                }
        }
return merge;
}


private void addTopicName(TopicName otherTN, Set scopeSet, Topic type, Set variantSet){
        TopicName newName =createTopicName(otherTN.getValue(),this,scopeSet);
        if(type != null){
                newName.setType(type);
        }
        if(!variantSet.isEmpty()){

                Iterator varIt = variantSet.iterator();
                while(varIt.hasNext()){
                //checks if the variant has an external or internal reference

                    Variant var = (Variant)varIt.next();
                    if(var.getResource()!= null){
                         newName.createVariant(var.getResource(),var.getScope());
                    }
                    else{

                            newName.createVariant(var.getValue(),var.getScope());
                    }
                }
        }
}
```

The creation of a topic map and how it is inserted into the filesystem is illustrated below,

```
stream = getClass().getResourceAsStream(
```

```
            IConstUtil.STR_XTM_AD_HOC_49);
```

topicMap1 = this.tmSystem.createTopicMap("http://www.joril.net/49_ad-hoc");

//the TopicMap consrtuctor is called, a new table is instantiated and filled up

```
ITree[] _oTree = new AVLTree[IMConst.N_TREE_TOTAL];

// Set the associations indexing tree structure
_oTree[IMConst.N_TREE_ASSOCIATIONSINDEX] =
            new AVLTree(CompareID.getInstance());
MAssociationsIndex.setInstance(
      _oTree[IMConst.N_TREE_ASSOCIATIONSINDEX], this);

/ Set the topics indexing tree structure
_oTree[IMConst.N_TREE_TOPICSINDEX] =
      new AVLTree(CompareID.getInstance());
MTopicsIndex.setInstance(_oTree[IMConst.N_TREE_TOPICSINDEX], this);

// Set the source locator of this topic map
_oTree[IMConst.N_TREE_SOURCETLOCATOR] =
            new AVLTree(CompareString.getInstance());
// Set the subject locator of this topic map
_oTree[IMConst.N_TREE_SUBJECTLOCATOR] =
            new AVLTree(CompareString.getInstance());
// Set the subject identifier of this topic map
_oTree[IMConst.N_TREE_SUBJECTIDENTIFIER] =
            new AVLTree(CompareString.getInstance());

// Set the topics id from the xtm file
_oTree[IMConst.N_TREE_TOPICID] =
            new AVLTree(CompareString.getInstance());
```

//the parser, maps every item in the XTM document to an instance in the correct AVLTree

```
parser = new XTMParser(stream, topicMap1);

parser.parse();
```

Appendix C the Merging Midlet

This is the code of the prototype.

```
/**
 * Midlet for testing the merging functionality
 * The different tests are:
 * <ul>
 * <li>Loading Topic Maps of different sizes</li>
 * <li>Merge 2 topic maps, limited</li>
 * <li>Deleting the topic maps</li>
 * <li>Complete merge</li>
 * <li>Displaying the contents of the merged topic map</li>
 * </ul>
 *
 * @author Joril Andersen
 * @version 1.0
 * @since 1.0
 */
public class Merge extends MIDlet implements CommandListener {

      /** A logger instance for getting information about the output */
   protected static final ILogger logger = Logger.getInstance();

   /** The 'EXIT' command */
   private final Command exit = new Command("Exit",   Command.EXIT, 1);


   /** The 'clear' command */
   private final Command clear = new Command("Clear screen", Command.ITEM ,
1);

    /** The 'read multiple XTMs' command */
   private final Command merge = new Command("Merge", Command.ITEM , 2);

   /** The 'loading' command */
   private final Command load = new Command("Load TMs", Command.ITEM , 1);

   /** The 'delete' command */
   private final Command delete = new Command("Delete topic map 2",
Command.ITEM , 2);

   /** The 'print' command */
   private final Command print = new Command("Print topic map",
Command.ITEM , 2);

   /** The 'print' command */
   private final Command printAssoc = new Command("Print associations",
Command.ITEM , 2);

   public XmlPullParserFactory xmlFactory = null;
   private TopicMapSystemFactory tmFactory =null;
   private TopicMapSystem tmSystem = null;
   TopicMap topicMap1= null;
   TopicMap topicMap2 = null;

   private Form form;
   private Display display;

   public Merge() {
```

```
        super();
        /*gets the display instance of the current midlet*/

        this.display = Display.getDisplay(this);
        if(Logger.getScreen() != null){
              form = (Form) Logger.getScreen();
              form.setTitle("The Merging Midlet\n\n **\\**");
         }
         else{
          /* runs on an emualtor or J2se */
              form = new Form("MTV -Test on emulator");

         }
        form.append(new StringItem("\n\t -TM4J2ME 1.0- \n\n 1.Load \n
        2.Merge\n" +
                          " 3.Print merged maps\n" +
                          " 4.Delete topic maps\n\n" +
                          "-By Joril Andersen", ""));

        form.addCommand(this.exit);
        form.addCommand(load);
        this.form.setCommandListener(this);
        this.display.setCurrent(this.form);
}

public void commandAction(Command command, Displayable displayable) {
        if(command== print){
              form.deleteAll();
              this.printMergedMap(topicMap1);
              form.addCommand(delete);
         }
         if(command ==printAssoc){
              form.deleteAll();
              this.printAssociations(topicMap1);
         }
        if (command == merge){
          this.form.deleteAll();

          form.removeCommand(merge);
          form.addCommand(print);
          form.addCommand(printAssoc);
          merge();

       }
        if(command == delete){
          this.form.deleteAll();
          deleteTopicMaps();
          logger.info("Topic maps deleted.");
          form.addCommand(load);
          form.removeCommand(print);
          form.removeCommand(merge);
          form.removeCommand(delete);
        }
        if (command == load){
          this.form.deleteAll();
          loading();
          form.removeCommand(load);
          form.addCommand(merge);
          form.addCommand(delete);
          form.addCommand(print);
          form.addCommand(printAssoc);
```

```
    }
    if (command == this.clear){

      this.form.deleteAll();
   }
    if (command == this.exit){
         try {
                destroyApp(false);
         }
         catch (MIDletStateChangeException e) {
                     e.printStackTrace();
         }
          notifyDestroyed();
    }
}
private void deleteTopicMaps(){
      topicMap2 = null;
      //topicMap1 = null;
      this.xmlFactory = null;
      this.tmSystem = null;
      cleanUp();
}

private void loading (){
   XTMParser parser = null;
   try{

    xmlFactory =XmlPullParserFactory.newInstance();
    xmlFactory.setNamespaceAware(true);
    tmFactory = TopicMapSystemFactory.newInstance();

    tmSystem = tmFactory.newTopicMapSystem();

    InputStream stream = null;
    long startTime = System.currentTimeMillis();

    stream = getClass().getResourceAsStream(
                          //IConstUtil.STR_XTM_AD_HOC_84);
                          //IConstUtil.STR_XTM_AD_HOC_49);
   IConstUtil.STR_XTM_TEST2);

   /* createTopicMap():
   * instantiates the index
   * instantiates the main tree with the smaller trees
   * add the topic map to the filesystem
    */
   topicMap1 = this.tmSystem.createTopicMap
   ("http://www.joril.net/49_ad-hoc");

   try{
   //The XTM document is deserialized and duplicates are
  //checked in the FS
         parser = new XTMParser(stream, topicMap1);
         try {
               parser.parse();
         } catch (IOException e) {
               logger.error("the parsing met a i/o exception");
               e.printStackTrace();
         }
   }
   catch(XTMParserException e){
```

```
                logger.error("The first topic map could not be parsed
        ",e);
                e.printStackTrace();
}
Set nrTopics = topicMap1.getTopics();
Set nrAssocs = topicMap1.getAssociations();
logger.info("1.Number of topics :"+nrTopics.size());
logger.info("1.Number of assocs :"+nrAssocs.size());
stream = getClass().getResourceAsStream(
        //IConstUtil.STR_XTM_AD_HOC_112);
        //IConstUtil.STR_XTM_AD_HOC_45);
        IConstUtil.STR_XTM_TEST1);

topicMap2 = this.tmSystem.createTopicMap
("http://www.joril.net/45_ad-hoc");

try{
        parser = null;
        parser = new XTMParser(stream, topicMap2);
        try {
                parser.parse();
        } catch (IOException e) {
                logger.error("the parsing of topic map 2 met a i/o
                exception");
                e.printStackTrace();
        }
                }
        catch(XTMParserException ex){
                logger.error("The second topic map could not be parsed
                ",ex);
                ex.printStackTrace();
        }
        Set nrTopics2 = topicMap2.getTopics();
        Set nrAssocs2 = topicMap2.getAssociations();
         logger.info("2.Number of topics :"+nrTopics2.size());
        logger.info("2.Number of assocs :"+nrAssocs2.size());
        try {
                stream.close();
                parser = null;
        } catch (IOException e) {
                logger.error("problems closing the stream..",e);
                e.printStackTrace();
        }
        long stopTime = System.currentTimeMillis();
        long runTime = (stopTime - startTime);
        logger.info("Time loading:"+Long.toString(runTime));

   }
catch(TMAPIRuntimeException e){
        logger.error("TopicMap api runtime exception!",e);
}
catch(OutOfMemoryError e){
        logger.error("Out of memory!",e);
}
catch(FactoryConfigurationException ex){
        logger.error("The config of the class failed!",ex);
}
catch(TMAPIException exx){
        logger.error("The config of the class failed!",exx);
}
catch(NullPointerException exx){
```

```
                logger.error("The object is null!",exx);
        } catch (XmlPullParserException e) {
                logger.error("XML pullparser unntak",e);
                e.printStackTrace();
        }
    }

// exploring the merging
 private void merge(){
        try{
          long startTime2 = System.currentTimeMillis();
          topicMap1.mergeIn(topicMap2);
          long stopTime2 = System.currentTimeMillis();
          long runTime2 = (stopTime2 - startTime2);

         logger.info("Time merging: "+Long.toString(runTime2));
        }
        catch(OutOfMemoryError e){
                logger.error("Out of memory!",e);
        }
        cleanUp();


    }
    /* returns the topic names as strings*/
public static String getTopicNamesString(Topic t) {
        String result = "";
        Iterator it = t.getTopicNames().iterator();
        while (it.hasNext()) {
                TopicName tn = (TopicName) it.next();
                Topic scope = (Topic)tn.getScope().iterator().next();
                if(scope != null){
                //TODO write out every scope, if many...
                TopicName scopeName = (TopicName)
                scope.getTopicNames().iterator().next();
                // if the topic does not have a name, check //item
                identifier/source locator
                        if(scopeName ==null){
                                Locator scopeLoc =(Locator)scope
                        .getSourceLocators().iterator().next();
                            result += tn.getValue()
                        +"(scope:"+scopeLoc.getReference()+")";
                        }
                        else{
                                result += tn.getValue() +"(scope:"
                                +scopeName.getValue()+")";
                        }
                }
                else{
                        result += tn.getValue()+"(no scope)";
                }
                if (it.hasNext()) {
                        result += ", ";
                }
        }
        return result;
}
/* the merged topic map is printed to screen */
private void printMergedMap(TopicMap topicMap){
        Set nrTopics = topicMap.getTopics();
        Set nrAssocs = topicMap.getAssociations();
```

```
logger.info("Number of topics :"+nrTopics.size());
logger.info("Number of assocs :"+nrAssocs.size());
Topic [] me = getTopicsArray(nrTopics);
for(int i =0; i < me.length; i++){
      Topic topic = me[i];
      logger.info("Topic :"+i+":"+getTopicNamesString(topic));//+

      Locator l = null;

      if((Locator)topic.getSubjectIdentifiers()
      .iterator().next() != null){
            l= (Locator)topic
            .getSubjectIdentifiers().iterator().next();

            if(l.getReference() != null){

                  logger.info("PSI:"+l.getReference());
            }
            if(l.getNotation() != null){

                  logger.info("PSI:"+l.getNotation());
            }
      }
      Locator sl = null;
      Iterator itSource = topic.getSourceLocators().iterator();
      while( itSource.hasNext()){
            sl = (Locator)itSource.next();
            if(sl.getReference() != null){
                  logger.info("Item identifier:"+sl.getReference());
            }
      }
      Locator subl = null;
      if( (Locator)topic.getSubjectLocators() .iterator().next() !=
      null){
            subl = (Locator)topic.
            getSubjectLocators().iterator().next();
            if(subl.getReference() != null){
                  logger.info("Subject Loc:"+subl.getReference());
            }
            if(subl.getNotation() != null){
                  logger.info("Subject Loc:"+subl.getNotation());
            }
      }
      Set sett =topic.getTypes();
      if( !sett.isEmpty()){
            Iterator it = topic.getTypes().iterator();
            while(it.hasNext()){
                  Topic topic2 = (Topic)it.next();
                  logger.info("Topic
                  type:"+getTopicNamesString(topic2));

            }
      }
      Occurrence oc = null;

      if((Occurrence)topic.getOccurrences()
      .iterator().next() != null){
            oc =(Occurrence)topic.
            getOccurrences().iterator().next();
            if(oc.getResource().getReference() != null){
                        logger.info("Occurrence:"+oc.
```

```
                        getResource().getReference());
                }
                if(oc.getValue()!= null){
                  logger.info("Occurrence:"+oc.getValue());

                }
            }
            Iterator rolesPlayed =topic.getRolesPlayed() .iterator();
            Set setRolesPlayed =topic.getRolesPlayed();
            logger.info("Nr of roles:"+setRolesPlayed .size());
            while(rolesPlayed.hasNext()){
                  AssociationRole role =
                  (AssociationRole)rolesPlayed.next();
                  Association assoc = role.getAssociation();
                  Topic type = assoc.getType();
                  if(type != null){
            logger.info("In Association:"
                     +getTopicNamesString(type));
                  }
            }

      }
      printAssociations(topicMap);
}

/* The existing associations are written to screen*/
private void printAssociations(TopicMap topicMap){
      int index = 0;
      Iterator it1 = topicMap.getAssociations().iterator();
      Set set = topicMap.getAssociations();
      logger.info("Number of associations :"+set.size() );
      while(it1.hasNext()){
        index++;
        Association assoc = (Association)it1.next();
        if(assoc != null){
            Locator loc1 = (Locator) assoc
            .getSourceLocators().iterator().next();
            if(loc1 != null){
              logger.info(" Assoc :"+loc1.getReference() );
            }
            else{
                  logger.info(" Assoc :"+index );
            }
            Set tmp =assoc.getAssociationRoles();

            if(!(tmp.isEmpty())){
              Iterator it = assoc.    getAssociationRoles().iterator();

              while(it.hasNext()){
                  AssociationRole assRole = (AssociationRole)it.next();

                  if(assRole != null){
                    Topic player = assRole.getPlayer();
                  if(player != null){
                        Locator loc = (Locator)player
                        .getSourceLocators().iterator().next();
                        if(loc != null){
                        logger.info("player"+loc
                              .getReference());
                        }
                  }
```

```
                    Topic role = assRole.getType();
                    if(role != null){

                      Locator loc2 = Locator)role
                    .getSourceLocators().iterator().next();
                       if(loc2 != null){
                          .info(" role : "
                          logger+loc2.getReference());

                      }

                   }
                }
            }
        }
        Topic t = (Topic)assoc.getType();
        if(t != null){
              Locator loc2 = (Locator)t
              .getSourceLocators().iterator().next();
              if(loc2 != null){
                    logger.info("Assoc type:"+loc2.getReference());
              }
        }
    }
  }
}
private Topic[] getTopicsArray(Set topicSet) {
        Topic result[] = new Topic[topicSet.size()];
        Iterator it = topicSet.iterator();
        Topic t;
        int i = 0;
        while(it.hasNext()) {
              t = (Topic) it.next();
              result[i] = t;
              i++;
        }
        return result;
}
protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
              // TODO Auto-generated method stub
}
protected void pauseApp() {
        // TODO Auto-generated method stub
}
protected void startApp() throws MIDletStateChangeException {
// TODO Auto-generated method stub

}
public void cleanUp(){
        System.gc();
}
}
```

Appendix C on CD-ROM

The Mobile Merging Midlet (Eclipse project)
The project is developed in Eclipse and it is possible to run the midlet on emulator if the Java ME plug-in is imported. The project can also be deployed and transferred to a mobile phone. Instructions can be found in the README.txt file.