TOPIC MAP VISUALIZATION

by

Kerziban MUMCU

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

July 2005
Istanbul, Turkey

APPROVAL PAGE

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Kemal FIDANBOYLU
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Atakan KURT
Supervisor

Examining Committee Members

....

....

....

....

....

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

Assist. Prof. Nurullah ARSLAN
Director

Date
July 2005

TOPIC MAP VISUALIZATION

Kerziban MUMCU

M.S. Thesis – Computer Engineering
July 2005

Supervisor: Assist. Prof. Atakan KURT

ABSTRACT

In this thesis, we developed a tool to visualize topic maps in XTM syntax using higraphs. The need for visualization has arisen from the need to explore Topic Maps (TM). Our tool allows users to visualize and navigate a topic map, provides useful operations on data, and presents summary information about data. A data model for TM called multilevel higraph based on higraphs, and type hierarchies are offered in this work, which allows different kinds of visualization of a topic map at different levels of abstractions. In our model, a topic map can be represented by a higraph. TM contains a set of entities (topics, associations, roles and occurrences) and type hierarchies of these entities. In this study, we present four kinds of views of a topic map each of which displays type hierarchies of one entity.

Keywords: Topic Maps (TM), Higraph, TM Visualization, TM Type Hierarchies

KONU HARİTALARI GÖSTERİMİ

Kerziban MUMCU

Yüksek Lisans Tezi – Bilgisayar Mühendisliği
Temmuz 2005

Tez Yöneticisi: Yrd. Doç. Dr. Atakan KURT

ÖZ

Bu tezde higraph modeli kullanılarak XTM yapısında oluşturulan konu haritalarının gösterimini gerçekleştiren bir uygulama geliştirildi. Konu haritalarının gösterimi ihtiyacı, konu haritalarını araştırma, inceleme ihtiyacından doğmuştur. Geliştirdiğimiz uygulama, kullanıcılara veri üzerinde kullanışlı işlemler ve özet bilgiler sunarak onların bir konu haritasını dolaşmalarını sağlar. Konu haritaları için higraph yapısı üzerinde çok katmanlı higraph veri modeli ve veri içerisinde tür hiyerarşisi oluşturulmuştur. Bu da bir konu haritasının değişik seviyelerdeki değişik şekillerde gösterimini sağlar. Modelimizde konu haritaları higraph ile gösterilir. Konu haritaları; konular, konular arasındaki ilişkiler, oluşlar ve roller ve bunların tür hiyerarşilerinden oluşur. Bu tezde bir konu haritasındaki dört temel verinin tür hiyerarşisi için dört farklı gösterim sunulmuştur.

Anahtar Kelimeler: Konu Haritaları, Higraph, Konu Haritaları Gösterimi, Konu Haritalarında Tür Hiyerarşileri

To the world

ACKNOWLEDGEMENT

I would like to thank my thesis supervisor Assist. Prof. Dr. Atakan KURT for his guidance and insight throughout the research.

Thanks go to the jury members Assist. Prof. Dr. Veli HAKKOYMAZ and Assist. Prof. Dr. Tahsin UĞURLU for their valuable suggestions and comments. I also express my sincere appreciation to the faculty member Mine MERCAN for her motivation throughout my MS study.

I express my thanks and appreciation to my family for their understanding, patience and support. Lastly, but in no sense the least, I am thankful to my friends and colleagues who made my stay at university a memorable and valuable experience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE

LIST OF SYSMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

TM	: Topic Maps
TT	: Topic Type
AT	: Association Type
RT	: Role Type
OT	: Occurrence Type
TTH	: Topic Type Hierarchy
TTV	: Topic Type View
ATH	: Association Type Hierarchy
ATV	: Association Type View
RTH	: Role Type Hierarchy
RTV	: Role Type View
OTH	: Occurrence Type Hierarchy
OTV	: Occurrence Type View

CHAPTER 1

INTRODUCTION

Topic Maps (TM) was first introduced for representation of finding-aids associated with books and documents such as indexes, table of contents, thesauri and so on. TM model presently is used to represent ontologies and provides a framework for the development of web applications by improving access to published information. A topic map contains structured data describing any resources in the world. First version of TM standard, ISO 13250 was released in 2000 (Techquila).

A topic map document consists of four main entities: *topics*, *associations*, *roles* and *occurrences*. Figure 1.1 shows a sample topic map document. A topic represents a resource that can be anything we can think. Associations represent relations between those resources. Each topic plays a role in the associations. They are called members of the associations. Occurrences connect topics with the things they represent. They refer some resources that have some information about the subjects of the topics. TM has other data stored in topic attributes or other sub elements like base name, variant, scope etc. Topics, associations and occurrences have types that are topics. Roles also are topics themselves. Therefore, we can say that TM has a hierarchical structure.

```
<topicmap xmlns:xlink="http://www.w3.org/1999/xlink">
   <topic id = "person" />
   <topic id = "man" instanceOf = "person" />
   <topic id = "woman" instanceOf = "person" />
   <topic id = "mother" />
   <topic id = "father" />
   <topic id = "child" />
   <topic id = "sibling"/>
   <topic id = "family"/>
   <topic id = "terry" instanceOf = "man" >
      <occurs type = "logo"
href="http://www.anywhere.com/~terry/me.gif" /> </topic>
   <topic id = "jane" instanceOf = "woman">
     <occurs type = "paper"
href="http://www.somewhere.com/~jane/papers.html" /> </topic>
   <topic id = "john" instanceOf = "man" >
   <topic id = "sue" instanceOf = "woman" >
   <topic id = "mary" instanceOf = "woman" >
<assoc instanceOf = "family" >
     <assocrl xlink:href="#jane" role="mother"/>
     <assocrl xlink:href="#terry" role="father"/>
     <assocrl xlink:href="#mary" role="child"/>
     <assocrl xlink:href="#john" role="child"/>
   </assoc>
<assoc instanceOf = "sibling" >
     <assocrl xlink:href="#john" role="sibling"/>
     <assocrl xlink:href="#mary" role="sibling"/>
   </assoc>
</topicmap>
```

Figure 1.1: A sample TM document

Since TM has attracted attention especially with the recent developments in ontology, semantic web and XML, there is a lot of interest for visualizing and navigating topic map documents. There are several approaches to visualize TM data. Some display data text-based while some uses 2D or 3D objects to display topics, associations and other data in a topic map (OLA, TM4J). However, visualizations using higher dimensions are also tried (Le Grand, Soto, Dodds, 2001).

Purpose of the information visualization is to help user to navigate and search large amount of data using graphical representation instead of visualizing data in text format. People are good at scanning and remembering images. Graphical elements facilitate representation and comparison of attributes in data via length, size, shape, color, orientation, texture etc. Information visualization has a principle: “Overview first, zoom and filter and then detail on demand” (Shneiderman, 1996). Visualization tools usually show an overview of data at first and enable user to quickly understand the data and discover important relationships without need for focusing attentions. In the second step, they allow user to zoom in data, filter irrelevant things and then get more information on significant data. We followed the same steps in our tool, too. We tried to summarize data and to provide some overviews using hierarchies.

In our study, we implemented a 2D graphical representation of TM using higraph model. Higraph is a graph definition, which combines hypergraphs with Venn diagrams. Graphs and hypergraphs represent a set of elements together with some special relations on them, while, Euler/Venn diagrams represent collection of sets together with some structural (i.e. set-theoretical) relationships between them (Harel, 1988). Higraphs represent both capabilities (Harel, 1988). Each set in a higraph is represented by a unique blob, completely with its own full contour. Each set has a label. The only identifiable sets are atomic sets. Atomic sets are those represented by blobs residing on the bottom level of the diagram containing no wholly enclosed blobs within. Empty spaces always represent nothing at all, except if it is the area of an atomic blob. In Figure 1.2 an example of higraphs is shown.

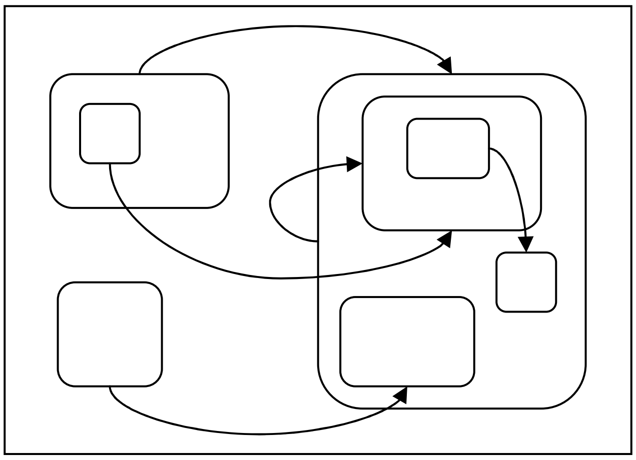


Figure 1.2: An example of higraphs

CHAPTER 2

BACKGROUND

2.1 TOPIC MAPS

Topic Maps (TM) is an abstraction for the representation and interchange of structured data. A topic map is an XML document that describes topics, the relationships between those topics and the occurrences of the topics. The basic idea underlying TM is to describe what an information set is about. TM declares topics and links the relevant parts of the information set to the appropriate topics. We can think of the topic map as a layer above information set as shown in Figure 2.1. Information sets can be resources like web pages, files, or other forms of information.

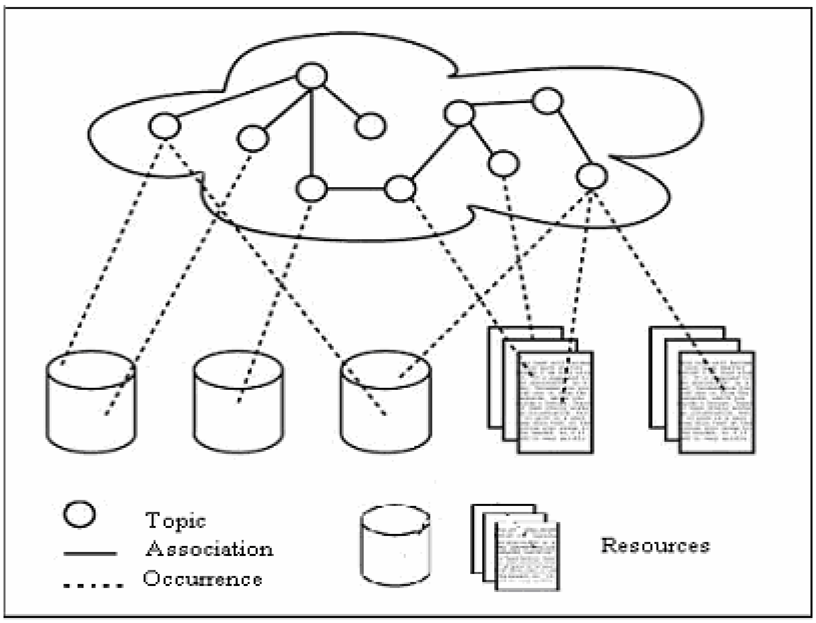


Figure 2.1: Topic Map and Information Set (Garshol, 2002)

Topics and associations among topics are basic structures in a topic map. Additional information is supplied in topics and associations.

2.1.1 Topics

Topics represent the things that the topic map is about. A topic can point to an electronic resource which can be retrieved by a computer or not, a real-world thing, an abstract concept etc. Figure 2.2 illustrates the representation of real world with topic maps.

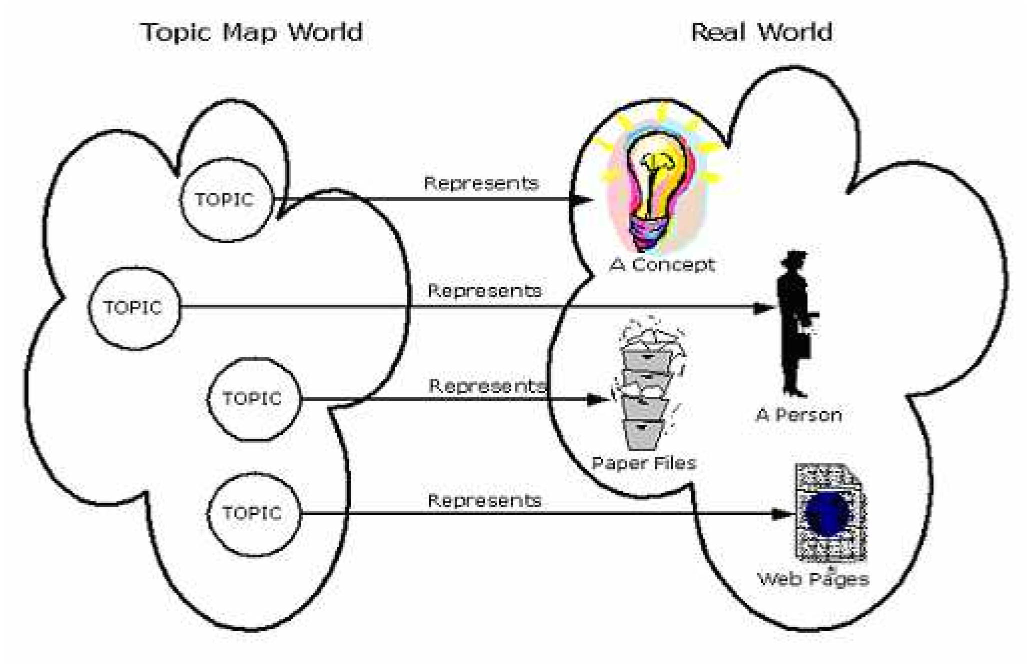


Figure 2.2: Representing the real world with topic maps (Techquila)

Topic Maps allow author to define some properties of topics. Those are the names, types, and occurrences of the topics. Actually, associations also contain information about topics but we will cover them in the following sections.

2.1.1.1 Names

A topic can have any number of names. At least one of them is base name. A name is a label for the topic, and can be used to display of the topic. Due to fact that a topic can have multiple names we can say that the names are synonyms or they are the labels have different representations which refer the same subject.

In TM, different topics can have the same name as different concepts may have same names in the real world. For instance, “letter” of the alphabet and the “letter” document can be defined as two topics in a topic map document or in different documents. Distinguishes of such topics can be done by other attributes like id, type, occurrences, and mostly by scope. We will mention those attributes in next sections.

2.1.1.2 Types

Using type attribute, we can indicate types of the topics. Types are also defined by topics. To declare a type for a topic, author creates a topic, which is a type, then creates the second topic and assigns the first one as the type of the second. Thus he/she says that the thing represented by second topic is_a thing represented by first topic. Let us simplify the concept with an example: Author creates two topics one is person and the other is “joe-strummer” with the type attribute pointing to the topic “person”. One can say that “joe-strummer” is_a “person”.

As a topic can have any number of types, a topic that is used as type of some other topics may also have own types since it is a topic.

Type attribute is very powerful and has important capabilities. One capability is to allow user to distinguish the topics which have same names but different type attributes and find the exact topic looking for. For example, the “glass” topic, which is a “cup”, can be differentiated from the eyeglass topic by having type attribute as pointing to topic “cup”.

In Figure 2.3 type topics, topics, occurrences and associations are represented.

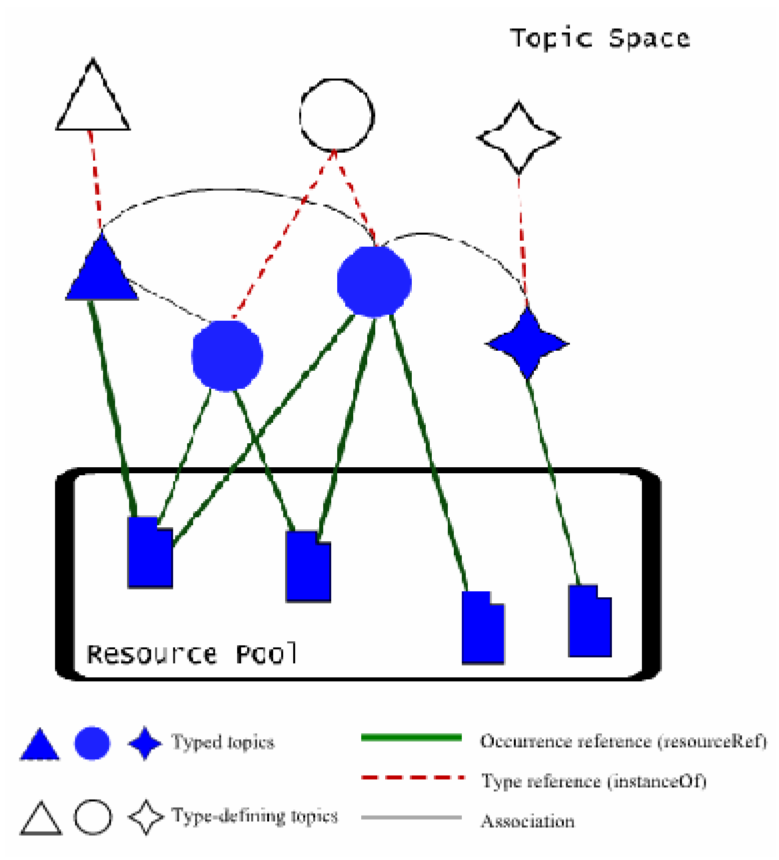


Figure 2.3: A topic map with topic types (Techquila)

2.1.1.3 Occurrences

Occurrences are the bridges between the topics and the things they represent. They point at some resources that have information about the subject the topic represent. The resources can be the subjects themselves. For example, topic representing the book “My Memories“ can have an occurrence indicating the webpage of the book “http://www.mywebsite.com/mybooks/mymemories.html” or a page that has some information about the book “http://www.books.com/newbooks.html”.

The resources can be retrievable or not however they are related to the topics in some way. Occurrences may have own types which are topics. For instances an occurrence which is a paper has the type “ printed “ while a logo occurrence has the type referenced to topic “image“. Different from the topics each occurrence can have only one type. Occurrences can be defined in specific scopes.

2.1.2 Associations

Associations represent the relationships between topics. An association can relate two or more topics. Like occurrences, associations can also have type. Types of an association express the kind of the relationship represented by the association. An example of associations is shown in Figure 2.4**.** As we said before an association relates one or more topics. Those topics are called members of the association and each plays a role in that association.

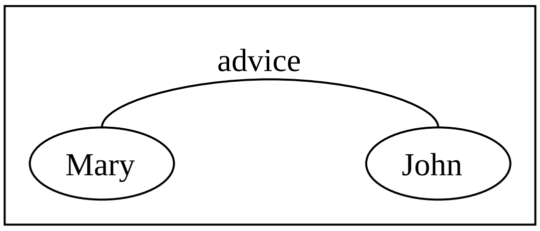


Figure 2.4: An example of association

2.1.2.1 Member and Role

An association consists of three kind of information: type of the relation, members of the relation and the roles of the members in that relation. For example assume that Mr. John and Terry are two person represented by two topics, John, Terry respectively. Let us say that John is the father of Terry. Then we create an association between those topics. The type of the association can be another topic “relative”. John and Terry are two members of the association. Then we associate a role “father” with the member “John” and role “child” with the member “Terry”. Thus, we said that there is a “relative” relation between topics “John” and “Terry”, “John” is “father” of “Terry”, “Terry” is “child” of “John”. Figure 2.5 shows this example association. As members are topics, roles also can be topics.

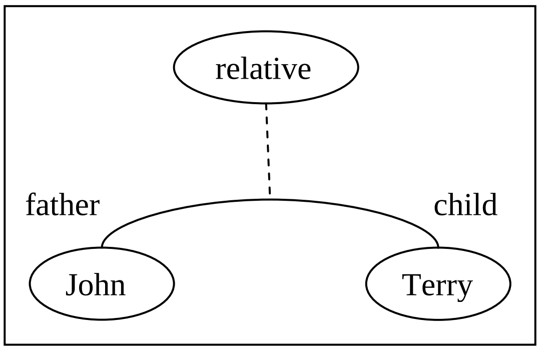


Figure 2.5: Example of an association with type and roles

2.1.3 Scope

Scopes specify contexts where some characteristics of topics are valid inside. Actually, everything in a topic map may be defined in scopes. One common use of scope is to support Multilanguage information. For example, we could create a topic representing “house” and give three names as “house” in English, “ev” in Turkish, and “haus” in German. Those names of topic have following scopes respectively, “en”, “tr”, and “de”. Thus, author can create a single topic to represent one concept and give three different names in different languages.

Scopes can also be topics. In the example above, “en” is a topic indicating English and “tr” is topic indicating Turkish and “de” is a topic indicating German language.

2.2 XTM

Topic Maps is an international standard ISO/IEC 13250 established in 2000. It defines the basic model and provides a SGML (Standard Generalized Markup Language) based syntax for that model. It uses HyTime (Hypermedia Time Based Structuring Language) to build a layer of abstract topics and relations between these.

Topicmaps.org consortium has adapted topic maps for use on the web after one year later that the first version of topic maps was released. XML topic maps (XTM) 1.0 specification is published in 2001 by topicmaps.org and it is accepted as second edition of ISO 13250. Full text of XTM DTD is given in appendix A. Because of XTM is one of the main topic maps syntax today, we used it as the syntax of topic maps in our thesis. In addition, it is supported by all topic maps tools. A sample XTM document is shown in Figure 2.6

```
<topicmap>
<topic id = "person" />
<topic id = "course" />
<topic id = "student">
  <instanceOf> <topicRef xlink:href="#person"/></instanceOf>
</topic>
<topic id = "teacher">
  <instanceOf> <topicRef xlink:href="#person"/></instanceOf>
</topic>
<topic id = "Sue">
  <instanceOf> <topicRef xlink:href="#student"/></instanceOf>
</topic>
<topic id = "Mary">
  <instanceOf> <topicRef xlink:href="#teacher"/></instanceOf>
</topic>
<topic id = "CS101">
  <instanceOf> <topicRef xlink:href="#course"/></instanceOf>
</topic>
<association>
  <instanceOf> <topicRef xlink:href= "#teaching" /></instanceOf>
  <member>
    <instanceOf> <topicRef xlink:href= "#mary" /></instanceOf>
    <roleSpec id= "#instructor" /></member>
  <member>
    <instanceOf> <topicRef xlink:href= "#CS101" /></instanceOf>
    <roleSpec id= "#course" /> </member> </association>
</topicmap>
```

Figure 2.6: Sample XTM document

2.2.1 Basic XTM Syntax

Figure 2.6 shows a simple XTM document consists of seven topics and an association. A typical topic map include topics, associations, and additional information like topic names, topic occurrences, roles that topics play, resource references and resource data etc. All information in a XTM document is stored in specific xml elements. Those elements are listed in the following table and each is discussed below.

ElementTag	Denotes
<topicMap>	Topic Map document element
<topic>	Topic element
<topicRef>	Reference to a Topic element
<subjectIndicatorRef>	Reference to a Subject Indicator
<instanceOf>	Points to a Topic representing a class
<subjectIdentity>	Subject reified by Topic
<baseName>	Base Name of a Topic
<baseNameString>	Base Name String container
<variant>	Alternate forms of Base Name
<variantName>	Container for Variant Name
<parameters>	Processing context for Variant
<association>	Topic Association
<member>	Member in Topic Association
<roleSpec>	Points to a Topic serving as an Association Role
<occurrence>	Resources regarded as an Occurrence
<resourceRef>	Reference to a Resource
<resourceData>	Container for Resource data
<scope>	Reference to Topic(s) that comprise the Scope
<mergeMap>	Merge with another Topic Map

Table 2.1: XTM Element Types (XML Topic Maps (XTM) 1.0, 2001)

2.2.1.1 <topicMap>

<topicMap> is the root element in XTM documents. Every XTM documents begins with <topicMap> element. Topics and associations are defined within this element. <topicMap> element defines the default namespace for xtm syntax and namespace for xlink. It may also have id and xml:base that is reference to document base URL attributes.

```
<topicMap xmlns='http://www.topicmaps.org/xtm/1.0/'
        xmlns:xlink='http://www.w3.org/1999/xlink'
        xml:base='http://www.somewhere.org/books/'>
<!-- topics, associations --></topicMap>
```

2.2.1.2 <topic>

Each <topic> element indicates a topic. It declares the names and the occurrences of a topic. A topic element may have zero or more names (<baseName>) and zero or more occurrences (<occurrence>). <topic> element may have zero or more <instanceOf> element and zero or more <subjectIdentity> element. <instanceOf> element defines the type of the topic. <subjectIdentity > element specify the subject identity of the topic. Each <topic> element may also have a unique id attribute.

```
<topic id="Sue">
   <instanceOf>
       <topicRef xlink:href="#person"/>
    </instanceOf>
   <!—other characteristics -->
</topic>
```

2.2.1.3 <topicRef>

It is a reference to a topic element. It provides a URI reference to a topic. The referenced topic can be defined in the same document or another one. <topicRef > element has attributes as follows: *id* a unique identifier, *xlink:type* identifies the link type, *xlink:href* gives the URI reference.

```
<topicRef xlink:href="#person"/>
```

2.2.1.4 <subjectIndicatorRef>

It is a reference to a subject indicator. <subjectIndicatorRef> is the same as <topicRef> except that it provides an URI reference to a resource. The resource is indicates the subject. It has the same attributes as <topicRef> element.

```
<subjectIndicatorRef
xlink:href="http://www.somewhere.com/books.html#mymemories"/>
```

2.2.1.5 <instanceOf>

<instanceOf> element specifies a type that the parent element belongs. It defines a class-instance relation between concepts. It has a child element either <topicRef> or <subjectIndicatorRef>. <instanceOf> element can have *id* attribute.

```
<topic id="mymemories">
   <instanceOf> <topicRef xlink:href="#book"/> </instanceOf>
</topic>

<topic id="mymemories">
   <instanceOf>
       <subjectIndicatorRef
xlink:href="http://www.somewhere.com/books.html"/>
   </instanceOf>
</topic>
```

2.2.1.6 <subjectIdentity>

It specifies the subject indicated by a topic. If the topic represent an addressable concept <resourceRef> element is used to address the concept. A topic may have only one resource that is considered as the subject of the topic.

If the topic has some resources indicating the subject, one or more <subjectIndicatorRef> elements are created as children of <subjectIdentity> element.

If the topic defines the same subject that indicated by another topic that topic can be referenced by <topicRef> child element. <subjectIdentity> may have *id* attribute.

```
<topic id="tr">
   <subjectIdentity>
       <subjectIndicatorRef
xlink:href="http://www.geography.com/countries/tr.html"/>
   </subjectIdentity></topic>
```

2.2.1.7 <baseName>

It specifies a name of the topic. A topic name is a string and it is the content of <baseNameString> element, <baseName> element has <baseNameString> element as a child. <baseName> may also have a <scope> element which specifies the context within the name is valid. If no <scope> child element exists it means that the name is always valid.

Another optional child element is <vairant> that is repeatable and provides alternative basenames.

```
<topic id="Terry">
   <baseName>
       <baseNameString>Terry Brown</baseNameString>
    </baseName>
</topic>
```

2.2.1.8 <baseNameString>

As we mention in <baseName> concept <baseNameString> element contains a string to represent the name of a topic. It is the child of <baseName> element. Its content is #PCDATA and it may have an *id* attribute.

```
<topic id="en">
   <baseName>
       <baseNameString>England</baseNameString>
   </baseName>
 </topic>
```

2.2.1.9 <variant>

<variant> element contains alternate form of a topic base name. It has a child <parameters> that specifies processing context that the alternative name is appropriate for. It may have a <variantName> element or not. <variantName> provides the alternative base name.

It may also have zero or more <variant> children to specify additional names for the topic. <variant> element may also have an *id* attribute.

```
<topic id="t-mick-jones">
  <instanceOf><topicRef xlink:href="#tt-musician"/></instanceOf>
  <baseName><baseNameString>Mick Jones</baseNameString>
    <variant>
      <parameters>
        <topicRef xlink:href="http://www.topicmaps.org/xtm/1.0/#psi-sort"/>
      </parameters>
      <variantName>
            <resourceData id="N129-N12f"> Jones, Mick </resourceData>
      </variantName>
    </variant>
    <variant>
      <parameters>
        <topicRef xlink:href="http://www.topicmaps.org/xtm/1.0/#psi-sort"/>
      </parameters>
      <variantName>
            <resourceData id="N129-N132"> jones mick </resourceData>
      </variantName>
     </variant>
  </baseName>
</topic>   (Pepper, music-xtm.xml)
```

2.2.1.10 <variantName>

It provides a base name in two ways. One is to provide a resource reference by having <resourceRef> element as a child as a variant of a base name or to include a resource having <resouceData> element as a child. <variantName> may have an *id* attribute. A sample <variantName> can be shown in the example of 2.2.1.9.

2.2.1.11 <parameters>

<parameters> element provides a processing context for descendant <variantName> elements of it’s parent <variant> element. It contains one or more <topicRef> element or one or more <subjectIndicatorRef> element. It may also have an id attribute. Sample parameter usage can be shown in the example of 2.2.1.9.

2.2.1.12 <association>

<association> element specifies a relation between two or more topics. It may have zero or one <instanceOf > element that specifies the class to which <association> belongs. We can say it is type of the association or the relation specified by the association.

An <association> element may have a <scope> element to specify the context within the association is valid. One or more <member> children exist in an <association> element to specify the topics belonging to that association. Each topic plays a role in association. `*id*` attribute can be defined for an <association> element.

```
<association id="assoc1">
  <instanceOf><topicRef                                    xlink:href="#advice"
xlink:type="simple"/></instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#undergradadvisor" xlink:type="simple"/>
    </roleSpec>
    <topicRef xlink:href="#mary" xlink:type="simple"/></member>
  <member>
    <roleSpec><topicRef xlink:href="#undergradstudent" xlink:type="simple"/>
    </roleSpec>
    <topicRef xlink:href="#john" xlink:type="simple"/></member>
</association>
```

2.2.1.13 <member>

A <member> element in an <association> element specifies a role using <roleSpec> which is played in this association and specifies the topics play the role using <topicRef> or <resourceRef> or <subjectIndicatorRef> elements. <roleSpec> element is an optional child element in the content of <member> element. <member> element may have an id attribute.

```
<association id="assoc2">
  <instanceOf><topicRef xlink:href="#sibling-of" xlink:type="simple"/>
  </instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#sibling" xlink:type="simple"/>
</roleSpec>
    <topicRef xlink:href="#sue" xlink:type="simple"/>
    <topicRef xlink:href="#terry" xlink:type="simple"/></member>
</association>
```

2.2.1.14 <roleSpec>

It specifies a role that is played in an association by a member using <topicRef> or <subjectIndicatorRef> elements. It may contain an id attribute. An example usage of <roleSpec> element can be shown in the example of 2.2.1.13.

2.2.1.15 <occurrence>

<occurrence> element specifies a resource that has information about a topic using <resourceRef> or <resourceData> child element. <occurrence> element may have an <instanceOf> child to provide the class or type of it and it may have a <scope> child element to specify a context within the occurrence is valid. It may also have an id attribute.

```
<topic id="jane">
  <instanceOf><topicRef xlink:href="#instructor"/></instanceOf>
  <occurrence>
    <instanceOf><topicRef xlink:href="#paper"/></instanceOf>
    <resourceRef xlink:href="http://www.somewhere.com/~jane/papers.html"
xlink:type="simple"/> </occurrence>
  <occurrence>
```

```
      <instanceOf><topicRef xlink:href="#webpage"/></instanceOf>
      <resourceRef xlink:href="http://www.somewhere.com/~jane"
xlink:type="simple"/></occurrence>
</topic>
```

2.2.1.16 <resourceRef>

It supplies an URI reference of a resource using xlink:href attribute. It may also have id and xlink:type attributes. Examples of <resourceRef> can be shown in the explanation of <occurrence> element.

2.2.1.17 <resourceData>

It provides the resource data. The content of the element is #PCDATA and it may have an id attribute.

```
<topic id="mary">
   <instanceOf><topicRef xlink:href="#person"/></instanceOf>
   <occurrence><instanceOf><topicRef                    xlink:href="#date-of-
birth"/></instanceOf>
      <resourceData>1968-05-01</resourceData></occurrence>
</topic>
```

2.2.1.18 <scope>

It specifies a context in which given information is valid. It can be defined for base names, occurrences and associations.

```
<topic id="England">
   <baseName><scope><topicRef xlink:href="#en"/></scope>
      <baseNameString>England</baseNameString></baseName>
   <baseName><scope><topicRef xlink:href="#tr"/></scope>
      <baseNameString>Ingiltere</baseNameString></baseName>
</topic>
```

2.2.1.19 <mergeMap>

<mergeMap> is used to merge the containing topic map and the topic map referenced through xlink:href attribute of <mergeMap>. xlink:href attribute contains the URI of another topic map. <mergeMap> element may have zero or more <topicRef> or <resourceRef> or <subjectIndicatorRef> children to indicate some topics that the scope information will be modified.

```
<mergeMap xlink:href="http://www.somewhere.com/books.xtm">
   <topicRef xlink:href="#mymemories"/>
</mergeMap>
```

CHAPTER 3

TM DATA MODEL

In this chapter, we present our data model for topic maps. We will explain topic maps, hypergraphs, higraphs and type hierarchies in a topic map. Algorithms to create those hierarchies will be given and views of the hierarchies are going to be described.

3.1 TOPIC MAPS

A topic map is a collection of topics (T) and associations (A) among topics. Topics have occurrences (O) and roles (R) that are played in associations. An association can be defined between 2 or more topics each of which plays a role in that association. By nature, a role is a type for a topic. For example, assume that “Ann” is a topic and plays “mother” role in an association. Then we can say that “Ann” is a “mother”. A type can be likened to a class in object-oriented paradigm. A type is a set of entities. We defined 4 kinds of entities for any topic map: topics, roles, occurrences, and associations. Topics, occurrences, and roles structurally similar however associations are fundamentally different from other three. There are 4 kinds of types for 4 kinds of entities: Topic Types (TT), Association Types (AT), Role Types (RT), and Occurrence Types (OT). Again, TT, OT, and RT are similar in structure, while AT has a different nature. There is no limitation for a type being a topic in original definition of TM. No clear distinction among TT, AT, RT, and OT is made either. A topic may be type of a topic as well as of an association, of an occurrence, or of a role.

The <instanceOf> child element of a <topic> element, of an <association> element, of an <occurrence> element or of a <roleSpec> element defines a sub/super class hierarchy in a topic map. Thus a simple type hierarchy based on the <instanceOf> relationship can be constructed for a topic map. We create four kinds of views of a topic map for TT, AT, RT, and OT hierarchies: Topic Type View (TTV), Association Type View (ATV), Role Type View (RTV), and Occurrence Type View (OTV).

Hierarchies can be represented by UML class diagrams. However a class diagram represents only the type hierarchy and does not represent the inter-object relationship corresponding to associations among topics in a topic map. In the following figures (Figure 3.1, Figure 3.2, Figure 3.3, Figure 3.4) some graphical representations of the TT, RT, AT and OT hierarchies of a sample topic map school.xtm are presented. Full document of the school.xtm can be found in Appendix B. Type hierarchies and views will be described in section 3.4 by detail.

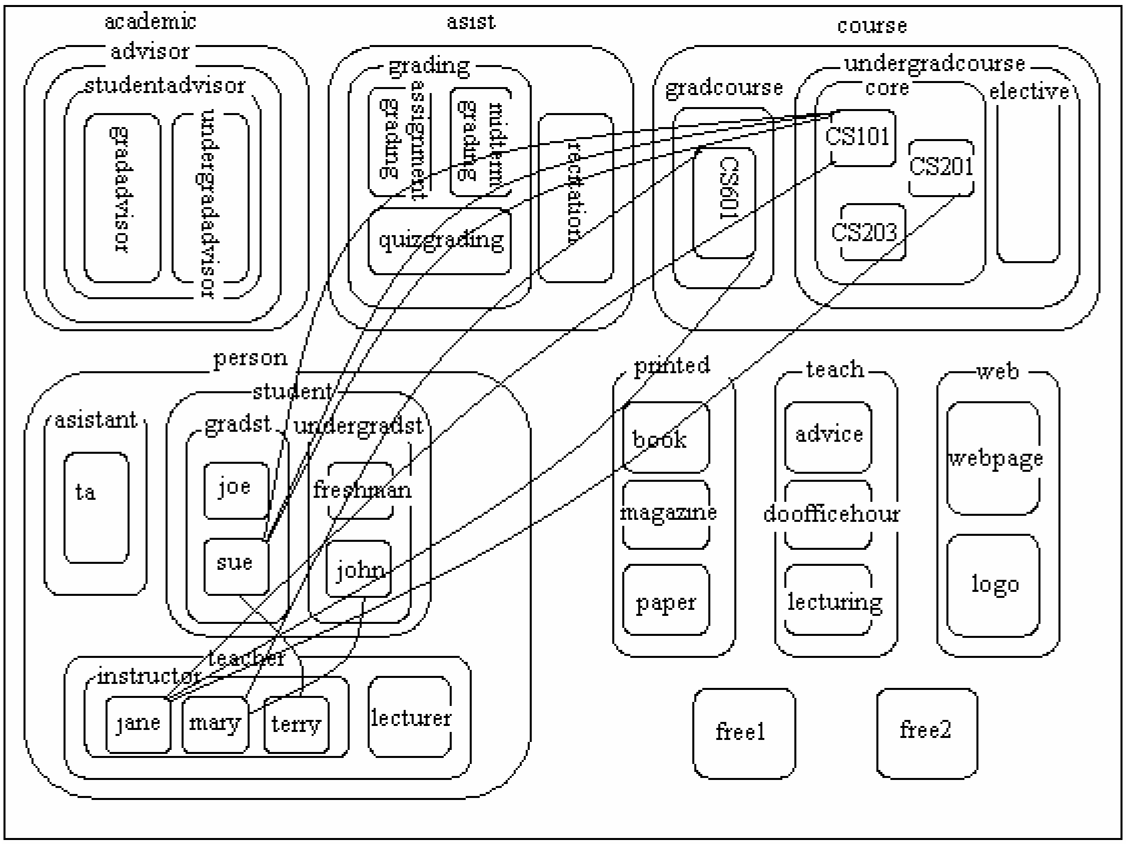


Figure 3.1: Topic type view

Figure 3.1 shows school.xtm topic map in which 49 topics exist and displayed as blobs. Blob labels are attached on blobs. In addition, nine associations among the topics are displayed by arcs. A blob represents the type topic of topics drawn inside it.

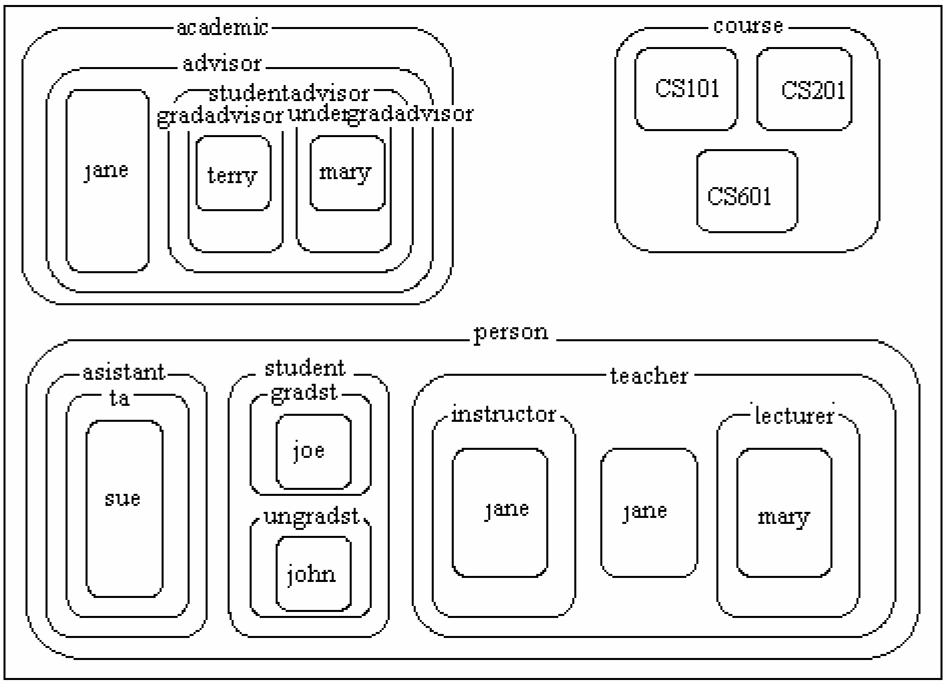


Figure 3.2: Role type view

Topics and roles played by those topics in school.xtm are displayed in the Figure 3.2.

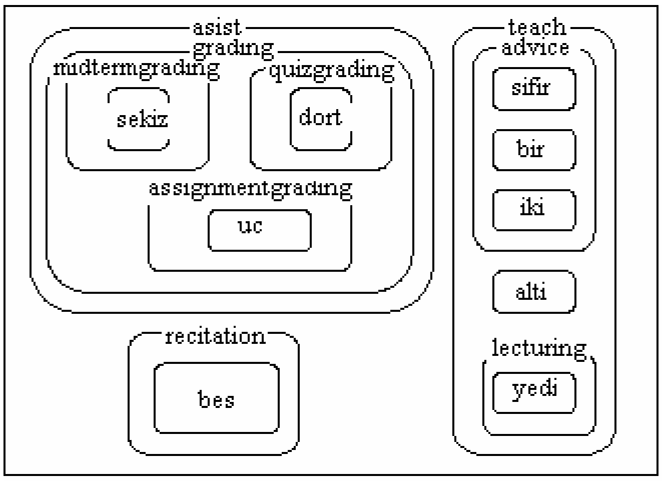


Figure 3.3: Association type view

Figure 3.3 shows association type hierarchy of school.xtm where associations are displayed by blobs as well as type topics. Associations are the blobs at the bottom level in the graph.

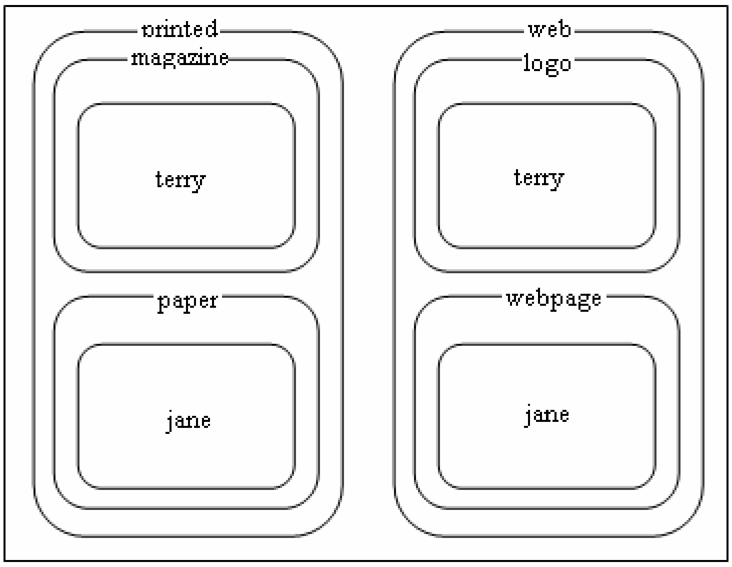


Figure 3.4: Occurrence type view

Topics and occurrences of those topics in the school.xtm are shown in the Figure 3.4. There are 8 topics: printed, magazine, paper, web, webpage, logo, terry and jane where magazine, paper, webpage and logo are occurrences types and terry and jane are the topics having those type of occurrences.

3.2 HYPERGRAPH

A hypergraph is graph consists of a set of nodes and a set of hyper edges. A hyperedge is an edge that connects two or more nodes as opposed to a simple edge in regular graphs. An example hypergprah is shown in Figure 3.5. We assume edges to be undirected. Although a directed edge can be used for representing instance_of relationship between two topics, representing types and topics with nodes may be misleading.

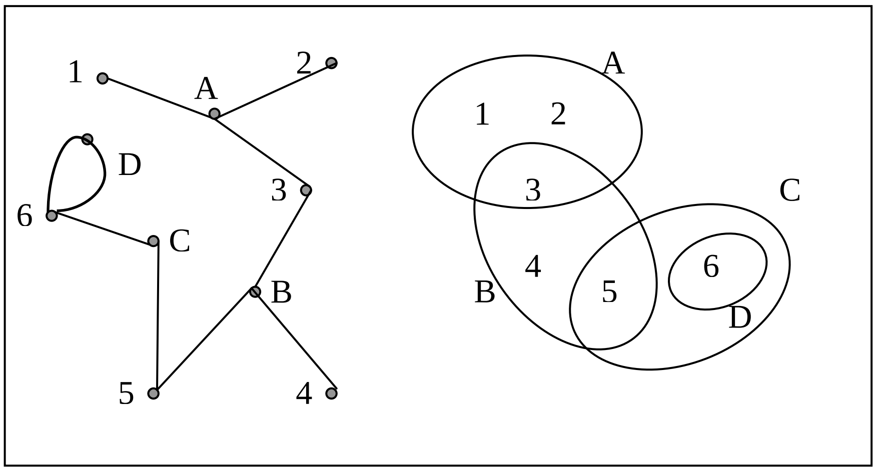


Figure 3.5: A graph and its corresponding hypergraph

A hypergraph can represent a TM where topics (1, 2, 3, 4, 5, and 6) are nodes and associations (A, B, C, and D) are hyperedges. Notice that roles in associations can be represented by labels next to topics. Alternatively, roles can be represented as nodes of different kinds. An example of topics and associations is depicted in Figure 3.6. Associations are represented by sets A1, A2, A3, A4 while topics are represented by their names, t1, t2, t3, t4, t5, t6. Though hypergraphs can represent a topic map, they can not properly represent a type hierarchy.

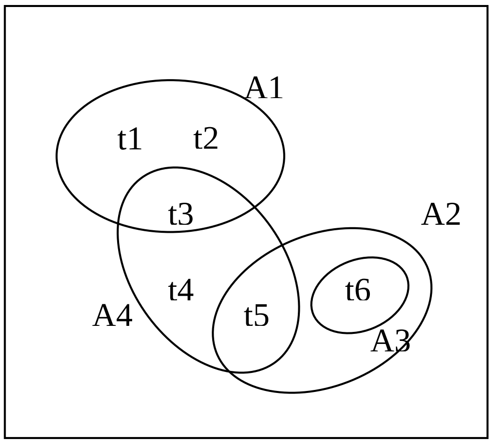


Figure 3.6: A hypergraph to represent topics and associations

3.3 HIGRAPH

Higraphs are an extension to hypergraphs. “The higraph, a general kind of diagramming object, forms a visual formalism of a topological nature” (Harel, 1988). A simple higraph is shown in Figure 3.7. Blobs are entities or sets. Each set has a label. Intersection of blobs indicates a set inclusion. Every set is denoted by a unique blob, complete with its own full contour. In the example below several sets are shown labeled by capital characters. Several cases of inclusion, disjointness and intersection of sets are shown in the Figure 3.7.

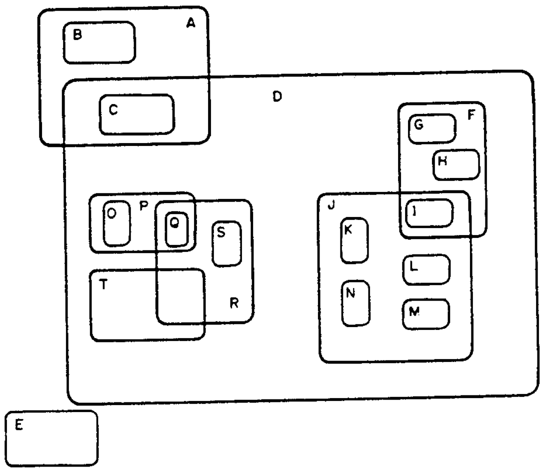


Figure 3.7: A sample higraph (Harel, 1988)

A higraph also contains hyperedges where a hyperedge connects blobs. Simply they are attached to the contour of any blob indicating that all the set denoted by that blob play in the relation denoted by the edge. As in hypergraphs, edges can be directed or not. A sample higraph containing an edge is shown in the Figure 3.8.

Any meaning can be attached to edges in higraphs. But the general meaning is that it represents a relationship between the sets it connects. Thus the edge in the figure means that “elements in the set A are related to some elements in the target set B”. We are not able to say that which elements of sets are related to some elements on the other sets even if edge is directed or not. Further information about higraphs can be found at (Harel, 1988).

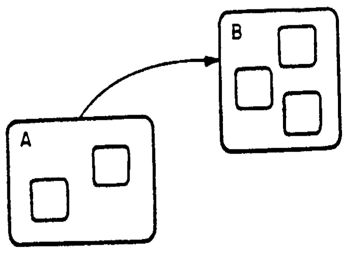
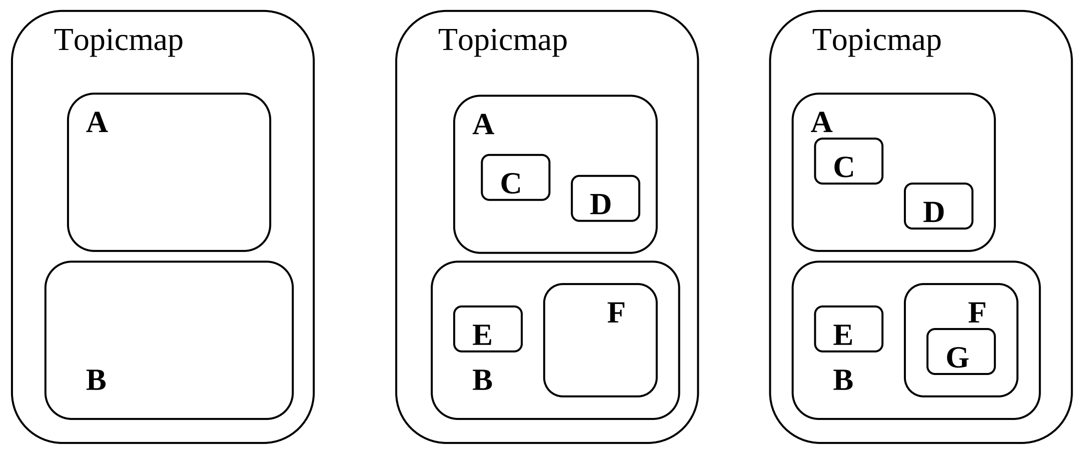


Figure 3.8: A simple higraph containing an edge (Harel, 1988)

Higraphs can be adapted to visualize topic maps such that blobs represent types in any type hierarchy and hyper edges represent associations. Blob containment, which represents set-inclusion, can be used to indicate type hierarchy (sub/super types). A blob is the type of the inner blobs. The only identifiable sets in a higraph are atomic sets, that is, those represented by blobs residing on bottom level of the diagram containing no wholly enclosed blobs within. We can use these bottom level blobs to correspond to topics that are not type of any topics in a topic map, that is, they do not contain any blobs. We can generalize the rule such that for any view TT view, AT view, OT view or RT view, blobs that contain at least one blob inside represent a type. Any blob that is empty represents an object (a topic, an association, etc).

3.3.1 Multilevel Higraphs

We added a new concept to higraph: multilevel higraph. A higraph can display a level of the hierarchies that a topic map has. Multilevel higraph is the collection of all the higraphs that display all levels in a hierarchy. For example, assume that we have a hierarchy of 3 levels. We can draw 3 higraphs to represent those levels. First higraph could display the root element in the hierarchy, while second displays root element and its children, and the third one displays the root, children of the root and the children of the children of the root. If the hierarchy is topic type hierarchy, then the higraph of the bottom level displays the all topics in the topic map. In Figure 3.9 an example of multilevel higraph for a topic map with an hierarchy of 3 levels is represented.



(a) Level 1: A higraph representing only the topics at top of TTH of a topic map

(b) Level 2: A higraph representing the topics at the first and second level of TTH of the topic map

(c) Level 3: A higraph representing an original topic map (all topics in TTH)

Figure 3.9: An example of multilevel higraph

Levels of a multi level higraph can be mixed according to blobs. For example content of blob B in the example above can be hidden at level 3 while other blobs are visible. In such a view, we can use blob levels instead of displaying static levels of the graph. We can attach level information to the blobs beginning with the root node that is in the level 0. Then we can represent blobs at different levels in the same view.

When a multi level higraph comes to mind, display of the relations in the original data must be redefined. If there is a relation among the objects that all of them are seen in the current higraph then we can represent that relation by an edge defined in the higraph concept. However, we have difficulty to display a relation at a level that some of its members are not seen.

We can find out several solutions. One is to display the relation at levels where all of its players are visible and do not display any relation if at least one of them is not visible. In this solution, we fail to see some relations at some levels. Thus, user cannot know the existing of relations of some visible blobs at a time. Another solution is to display the relation in partial such that to draw an edge between the visible players if at least two players are in the display. This solution may confuse users. Such as, an edge connecting two blobs can both represent a relation between two players or a relation between more players when the other players are in a different layers and not shown in the display.

Our solution is to display a relation when all of its players are visible as in the first solution, however, display metaedges for a relation if at least one of its players is not in the display. We call the original relations in the data as base relations and the edges as base edge. A metaedge represents a relation in a level for any relation that is visible at any lower level. Its players are players of the base relation that are visible if any exists and the ancestor blobs of players of the base relation that are not visible. It shows that there is a base relation between those blobs or some descendant blobs of those blobs. As a result, there will be a metaedge in any level for single base edge.

Afterwards, we modified our solution. In our first solution, there could be more than one metaedge between same blobs to represent some base edges. We decided to create only one metaedge between any blobs. A metaedge does not indicate a base edge any more. It indicates one or more base edges. An example of such metaedges is shown in Figure 3.10. First higraph contains seven blobs and three base edges between some blobs. In the second one, children of blob A and children of blob F are hidden, so the edges connected to that hidden blobs are also hidden. As a replacement for them, a metaedge between blob A and E and another metaedge between A and F are drawn. First metaedge is created for the metaedge between C and E and the second is created for two metaedges, one was between C and G and the other was between D and F. At the last higraph, we see only one metaedge indicating that there are some edges between children of A and children of B.

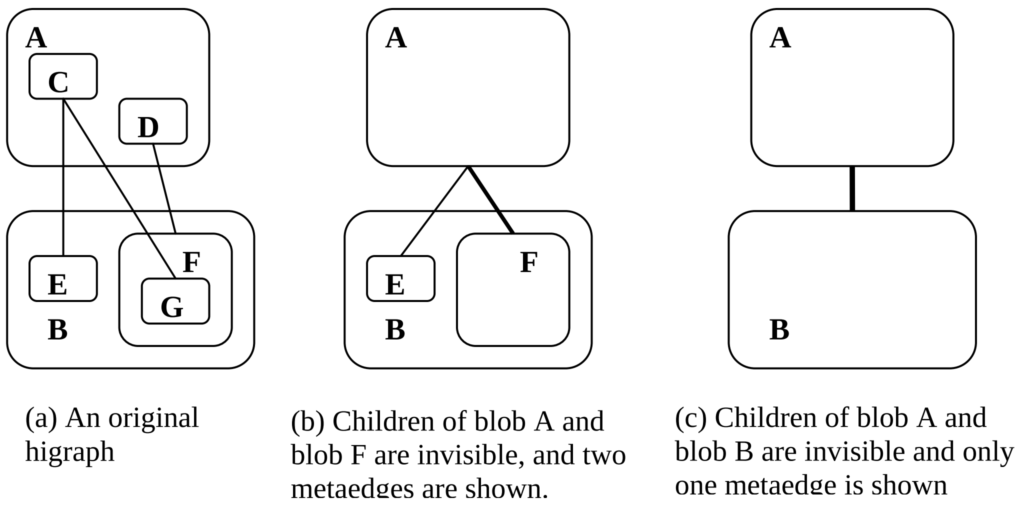

A
C
D
F
E
G
B
(a) An original higraph
A
E
F
B
(b) Children of blob A and blob F are invisible, and two metaedges are shown.
A
B
(c) Children of blob A and blob B are invisible and only one metaedge is shown

Figure 3.10: Examples of metaedges

As shown in the Figure 3.10 a hierarchy among edges is appeared. We know that "there is one metaedge between any blobs if there is at least one base edge among descendant of them". Consequently, we can simply say that a metaedge between any blobs is the ancestor of the edges (either meta or base) among the descendant blobs. Actually, this is only one of alternatives to create metaedges. Some alternatives are:

a) We can create one metaedge between any members for all edges among descendant members as shown in Figure 3.11. This is the alternative that we chosen to implement.

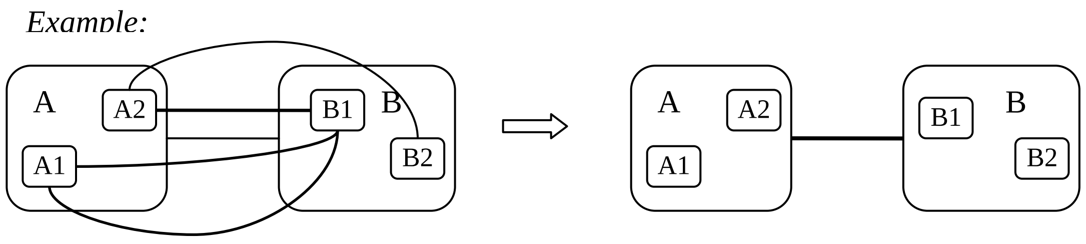


Figure 3.11: An example metaedge for any descendant edges

b) Edges could be grouped according to member blobs. That is if there are more than one edge between any members we will create only one parent metaedge for them.

Figure 3.12 shows an example where arcs between different members have different styles.

Example:

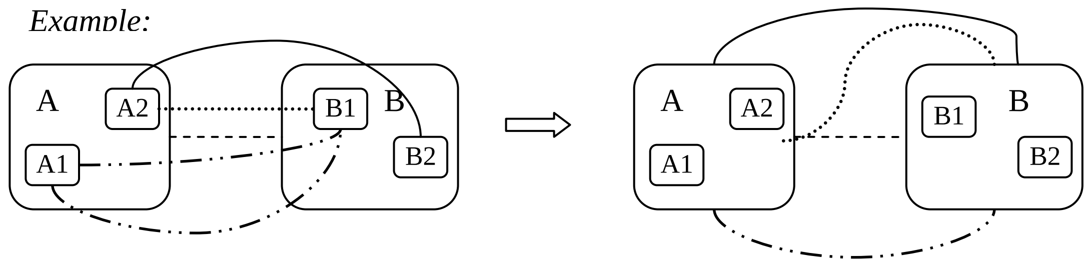


Figure 3.12: Example metaedges for each descendant edge grouped by member nodes

c) One metaedge can be created for the edges having the same type. Type of an edge is simply is the type of the association or relation it represents. An example is given in the Figure 3.13. Arc style represents arc types in the example.

Example:

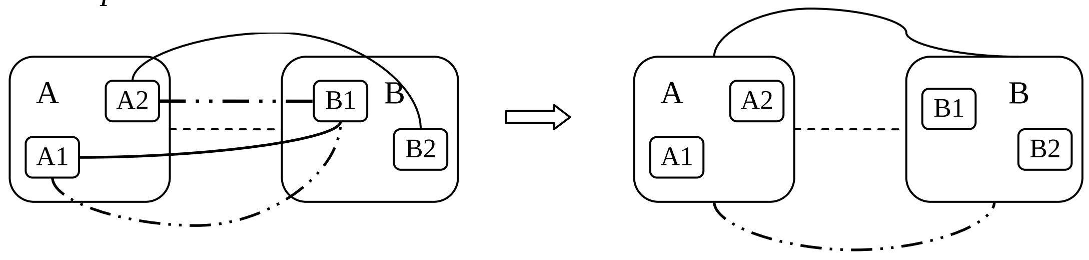


Figure 3.13: Example metaedges for any descendant edges grouped by relation type

d) Some combinations of the alternatives can be used. For example second and third alternatives could is combined in the following example. Accordingly, edges are grouped by both member nodes and types. If there are relations of the same type between same members one metaedge is created for them. Figure 3.14 illustrates an example of this alternative.

Example:

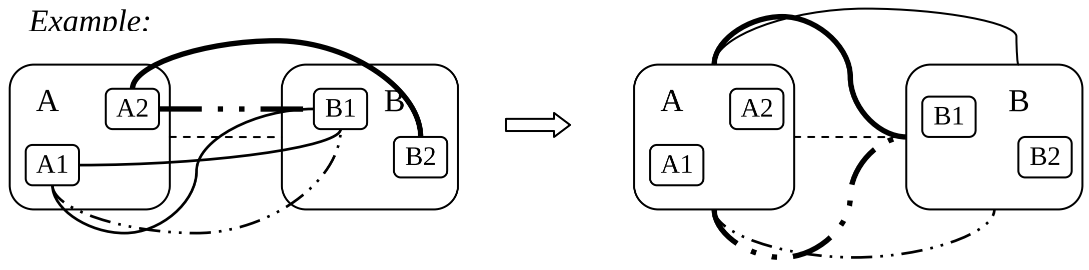


Figure 3.14: Combination of second and third alternative

Metaedgepart:

Metaedgepart is another type of edge that we constructed in multilevel higraphs. Metaedgeparts are created between parent blobs and children. A metaedgepart is an edge representing that there is a metaedge going out from parent that have an invisible child edge connected to that child blob. Metaedgeparts are connected to contour of parent blob at the same location with the related metaedge or metaedgepart outgoing from the parent. Metaedgeparts help user to estimate which children have relations between other blobs. A metaedge only states that there are relations between children of the parent with some blobs outside. However, it does not give any information about which children has relations. Metaedgeparts tell this information. Following figure shows an example of metaedgeparts. Metaedgeparts are represented by solid lines.

Example:

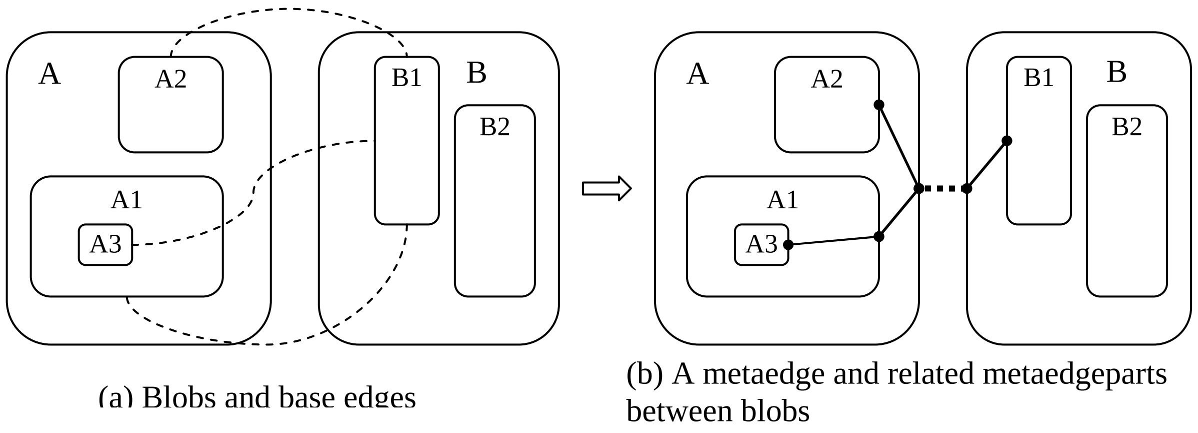


Figure 3.15: An example of metaedgeparts

In the next section, type hierarchies and the representations of hierarchies using higraph are explained.

3.4 TYPE HIERARCHIES

As we explain in the section 3.1, there are four kinds of types for four kinds of entities in a topic map. Hence, there are four hierarchies. Here we will give details of each hierarchy and their views.

3.4.1 Topic Type Hierarchy (TTH) and Topic Type View (TTV)

There are class-instance relationships between topics in a topic map provided by <instanceOf> element. The topic referenced in the content of an <instanceOf> element that is child of a <topic> element is a type for the topic indicated by that <topic> element. A topic may have zero or more types and a type topic could have own types as well. We call a topic base topic if it is not defined as a type in the topic map. When a topic is defined as a type is called type topic. All topics in a topic map that have no type topic are assigned to a default topic type for ease of presentation and manipulation.

In topic type hierarchy, we display topics, associations and the hierarchy among the topics. Topics are represented by blobs. A distinction can be made visually to differentiate among ordinary topics, and topics that are defined as roles or occurrences. Each kind of topics can be represented by a different blob shape. Although type topics and base topics are both represented by blobs, it is possible to represent different types of topics by different types of blobs if needed. For example, blob contour or color can be used for this purpose. Alternatively, one can display the type topics by blobs and base topics by dots in the blobs. Type hierarchy is symbolized by blob containment.

ISO 13250 Topic Maps syntax allows topics to have zero or more types. Multiple super-types (multiple inheritances) can be represented by intersection of corresponding blobs. In our work, we do not support multiple inheritances.

A type hierarchy including only topics types and relationships among them (associations) is called Topics Type Hierarchy (TTH). Algorithm to create TTH of topic maps is given in Figure 3.16. A topic type view (TTV) provides different levels of abstractions of TTH to the user for the explorations of topic maps. TTV is based on higraph notation with metaedges, which allows user to manipulate the TM visually. A set of operations is provided for visual manipulations such as filtering, zooming.

```
createTopicTypeHierarchy(TM):
Input: TM document
Output: Topics and associations
Algorithm:

  Topic [] topics;
  Association [] associations;

  parse(TM, topics, associations);
  createMetaAssociations(topics, associations)


parse(TM, topic, association):
Input: TM document
Output: Topics and associations
Algorithm:

  previousTopic = “”; occurrence= false; inRoleSpec = false; currentRoleNode = null;

  For each beginning element tag in the TM document
  If element is

    <topic>: If a node is not created for this topic before add a new node to the topics
       vector with type attribute is set to “topic”

    <instanceOf>: previousTopic=”instanceOf”

    <topicRef>:
      If previousTopic == “instanceOf”
          If occurrence == true //descendant of an occurrence element
              Create a node for the topic referenced by topicRef if it is not created before
              Put the node in occurNodesTable
          If parent instanceOf element is child of a topic element
              Create a type node for the topic if it is not created before
     Set node type of the type node as “type”
              Create parent-child relationship between type topic and the current topic
          If parent instanceOf element is child of an association element
              Set current association type as the topic referenced by topicRef
      Else if inRoleSpec == true
          Create a node for the topic referenced by topicRef if it is not created before
```

```
            Set node type as “role”
            Add this node into the roleNodesTable also
            Set currentRole Node as this node
         Else if previousTopic == “member”
            Create a node for the topic referenced by topicRef if it is not created before
            Add this node into the member nodes of the currentRoleNode
            Create a binary association between member node and currentHyperNode
            Add this arc into the arcs of the currentHyperNode and into the general arcs table

      <occurrence>: Set occurrence as true

      <association>: Set previousTopic as “association”
                    Create a new empty association as currentAssociation
                    Create a new node as currentHyperNode ans set node type as “hypernode”
                    Add the currentHyperNode into the nodes table and hypernodes table

      <member>: Set previousTopic as “member”
                    Create a new empty node as currentRoleNode and set type as “role”

      <roleSpec>: Set inRoleSpec as true

   For each end element tag in the TM document
      If element is
         <topic>: Set currentNode as null
         <association>: Set currentAssociation as null
         <roleSpec>: Set inRoleSpec as false
         <member>: Set currentRoleNode as null;
         <occurrence>: Set occurrence as false;
```

CreateMetaAssociations(NodesTable, AssociationsTable)

Input: Topics and Associations
Output: Metaassociations, Hypernodes and Metaarcparts
Algorithm:

```
   For each association in the Associations
      Get the member topics of the current association

      If this is a baseassociation
         If no metaassociation is found between member topics
            Create a metaassociation between members
            Create child-parent relation between baseassociation and created
               metaassociation where metaassociation is parent
            Set current association as created metaassociation
         Else
            Set current association as found metaassociation

         Put parent topics of the members into parentTopics
         ChangeParentsAndLevels(parentTopics, current association)
```

```
            While the level numbers of new parent topics bigger than 1
            If there is a metaassociation between parent topics
              Create child-parent relation between current association and found
                metaassociation where metaassociation is parent
              Create Metaarcparts between parent topics and member topics
            Else
              Create a metaassociation between parents
              Create child-parent relation between current association and created
                metaassociation where metaassociation is parent
              Create Metaarcparts between parent topics and member topics
              Set current association as new metaassociation
      Set member topics as the parent topics

      ChangeParentsAndLevels(parentTopics, current association)
      If parentTopics is empty break

    assignArcWeights()
```

ChangeParentsAndLevels(parents,association)

Input: Parent topics and current association
Output: new parent topics
Algorithm:

```
    Find the parent with the highest level number
    nextlevel is the highest level number minus 1
    If nextlevel < 1
      Empty the parent topics
    Else
      For each topic in the parent topics with the level number is equal to the highest
    level
            Remove the topic from parent topics
            Add the parent of the topic into the parent topics
```

assignArcWeights()

Input: hypernodes and metaassociations
Output: metaassociations with arc wieghts
Algorithm:

```
    maxNumberOfDescendants = 0

    For each hypernode hnode
      Nod = calculateNumberOfDescendants(hnode)
      If maxNumberOfDescendants < Nod
            maxNumberOfDescendants = Nod
    weightFactor = maxWeight /maxBNumberOfDescendants

    For each hypernode hnode
      Nod = hnode.getCustomizedData("numberOfDescendants")
      Arcs = hnode.getArcs()
      For each arc a in Arcs
           Set arc weight as (Nod+1)*weightFactor
```

calculateNumberOfDescendants(hypernode)

Input: a hypernode
Output: number of descendant hyperedges
Algorithm:

```
    If hyppernode.getCustomizedData("numberofchildren")!= null
      Return  hyppernode.getCustomizedData("numberofchildren")

    Set children as hypernode.getCustomizedData("children")

    If(children == null)
      hyperode.setCustomizedData("numberOfDescendants",0)
      return 0
    Else
      For each hypernode hnode in children
           Nod += 1+ calculateNumberOfDescendants(hnode)
           hyperode.setCustomizedData("numberOfDescendants",Nod)
           return Nod
```

Figure 3.16: Algorithm to create TTH

An example TTH and TTV is shown in the following figure. There are 10 topics and 4 base associations. Only base association represented in TTH and TTV. Label of the type topics are attached on the blobs, while base topic labels are written in the blobs in sample TTV.

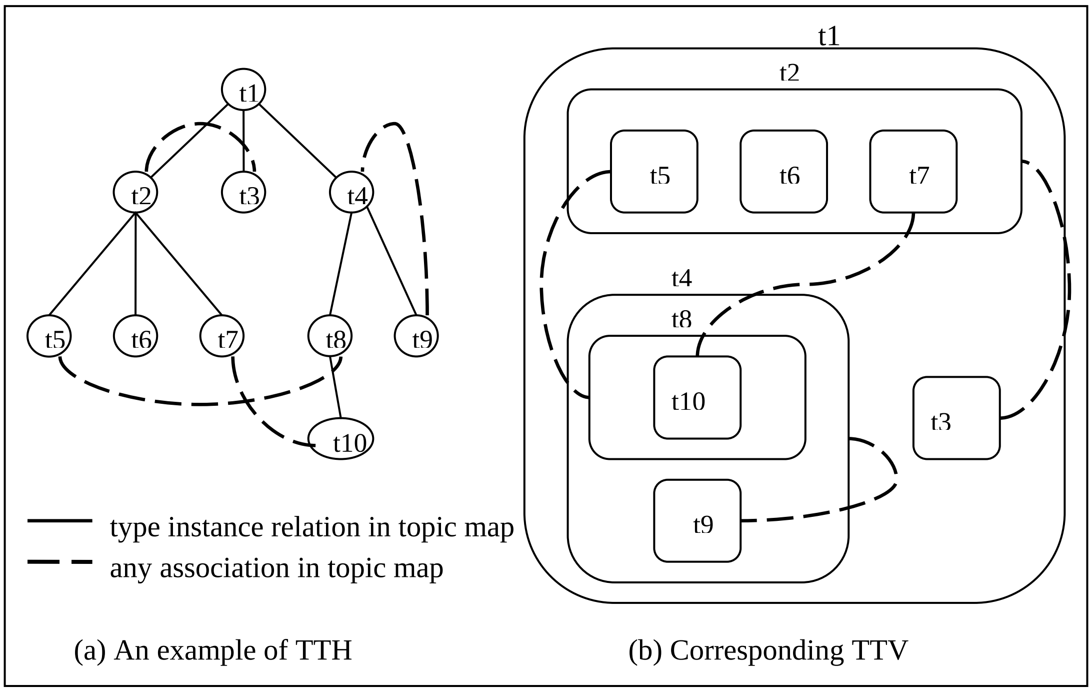

t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
type instance relation in topic map
any association in topic map
(a) An example of TTH
(b) Corresponding TTV

Figure 3.17: Example of TTH and TTV

3.4.2 Association Type Hierarchy (ATH) and Association Type View (ATV)

Association type hierarchy of a topic map includes the associations and the type topics of those associations with their own ancestor types. Topic maps syntax allow any <association> element to have an <instanceOf> child element so that an association can have a type. If no <instanceOf> child element is supplied for association it has the default association type.

Associations are treated as topics in association type hierarchy. They are represented by blobs like topics. If an association has a type then the type topic and its ancestor types up to root topic in topic type hierarchy are added to the association type hierarchy. Associations are the leaves in the hierarchy. Topic blobs and association blobs can be differentiated by node colors or shapes.

There is no hyperedge in the association type view. Type hierarchy is represented by blob containment.

Algorithm of the creation of association type hierarchy is given in Figure 3.18. Subsequent figure illustrates the same topic map in the example of topic type hierarchy with presenting association types, its ATH and ATV.

createAssociationTypeHierarchy(hypernodesTable, topicsTable)

Input: all topics in the topic map and hypernodes created in TTH
Output: ATH
Algorithm:

```
String file = "<?xml version=\"1.0\"" +
                    " encoding=\"UTF-8\" " +
                    " standalone=\"yes\"?>" +
                    " <assocTypeHierarchy " +
                    " xmlns=\"http://www.topicmaps.org/xtm/1.0/\" " +
                    " xmlns:xlink=\"http://www.w3.org/1999/xlink\">";

For each hypernode hnode in hypernodesTable
    Set type as hnode.getCustomizedData("associationType")
    Add "<topic id=\""+hnode.getNodeLabel()+"\"><instanceOf> " +
          "<topicRef xlink:href=\"#"+type+"\" xlink:type=\"simple\" />" +
          "</instanceOf></topic>" in file string
   parent = findNode(type)
   While parent != null
          Add parent node into nodesT
          parent = parent.getCustomizedData("parent")

For each node node in nodesT
   parent = node.getCustomizedData("parent")
   If parent != null
          Add "<topic id=\""+node.getNodeLabel()+"\"><instanceOf> " +
                    "<topicRef xlink:href=\"#" +parent.getNodeLabel()+
                    "\" xlink:type=\"simple\" /></instanceOf></topic>" in file string
   Else
           Add "<topic id=\""+hnode.getNodeLabel()+"\"/>" in file string

Add "</assocTypeHierarchy>" in file string
Write file into an xtm file.
```

Figure 3.18: Algorithm to create ATH

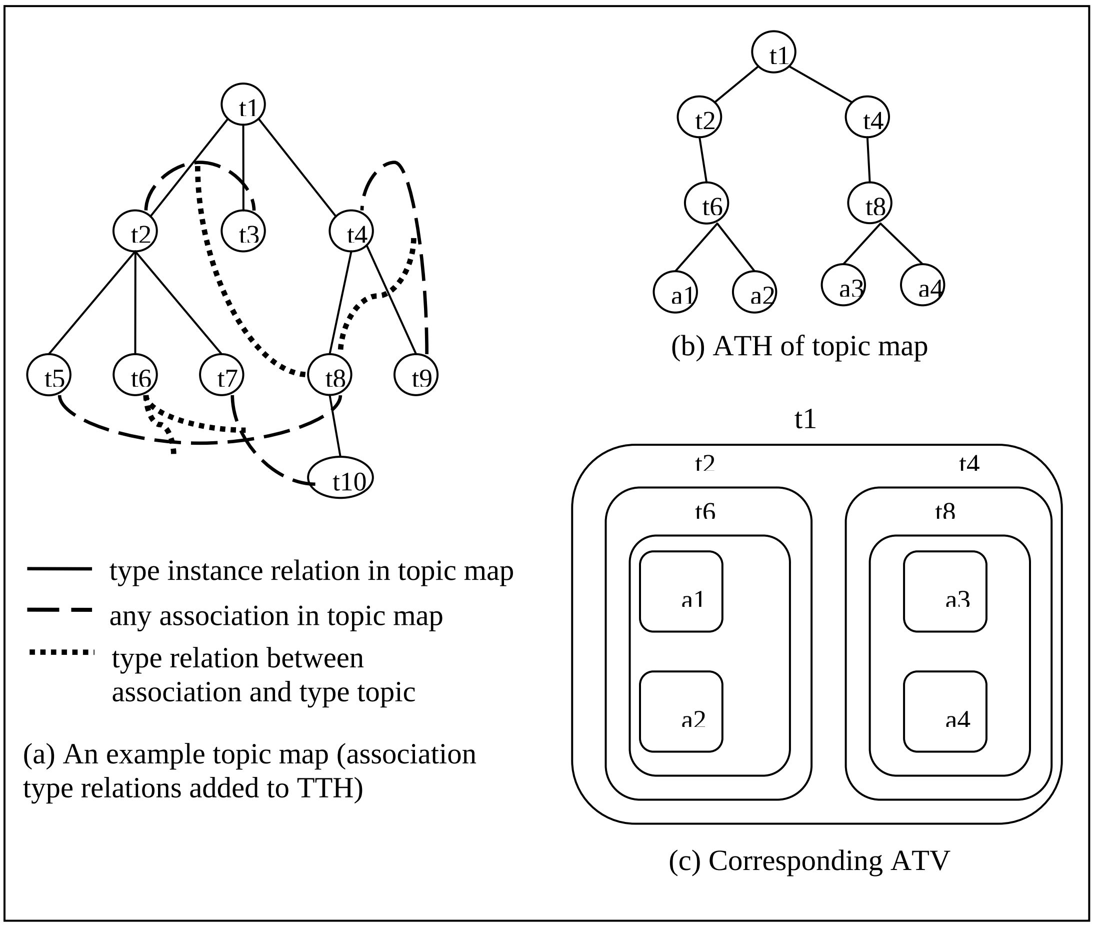

t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
type instance relation in topic map
any association in topic map
type relation between association and type topic
(a) An example topic map (association type relations added to TTH)
t1
t2
t4
t6
t8
a1
a2
a3
a4
(b) ATH of topic map
t1
t2
t4
t6
t8
a1
a2
a3
a4
(c) Corresponding ATV

Figure 3.19: Example of ATH and ATV

3.4.3 Role Type Hierarchy (RTH) and Role Type View (RTV)

Topics play roles in associations. A role can be a topic that is specified by content of <roleSpec> child element of an <member> child in an association. A role is considered as a type of its players in RTH.

A topic can exist in several associations and can play several roles in a topic map. Each role played by a topic becomes its type in RTH. If the role topics have own types then those type topics are also added to the hierarchy up to root node in the topic type hierarchy. If a role topic has not own type then it is assumed that it has default role type. Role type view displays topics and the roles which they play and the types of roles.

Role types are represented by blobs like topics in role type view. The hierarchy is represented by blob containment as well. Topics that play roles are at the bottom level of hierarchy. As a result of a topic can play several roles, it can be displayed under multiple role blobs. Following algorithm explains the role type hierarchy creation. Figure 3.21 shows an example of RTH and RTV.

```
createRoleTypeHierarchy(roleNodesTable, topicsTable)

Input: all topics in the topic map and role nodes found in TTH
Output: RTH
Algorithm:

Create a root element with the label “roleTypeHierarchy”

For each node rnode in roleNodesTable
  Add rnode into nodesT
  Set members as rnode.getCustomizedData(“members”)

  For each node member in the members
       If the member topic is not created before create a “topic” element
       If no “xlink:href”attribute equals to rnode.getNodeLabel() exists in descendant
       “topicRef”   element
              Create a “instanceOf” child for the “topic” element
              Create a “topicRef” element as child of the “instanceOf” element
              topicRef.setAttribute("xlink:href","#"+rnode.getNodeLabel())
       Append topic element as a child of the root element

       parent = rnode.getCustomizedData(“parent”)
       While parent != null
              add parent node into nodesT
              parent = parent.getCustomizedData(“parent”)

  For each node node in nodesT
       parent = node.getCustomizedData(“parent”)
       If element for this topic is not created before
              Create a “topic” element
       If parent != null and no “xlink:href”attribute equals to parent.getNodeLabel() exists in
       descendant “topicRef” element
              Create a “instanceOf” child for the “topic” element
              Create a “topicRef” element as child of the “instanceOf” element
              topicRef.setAttribute("xlink:href","#"+parent.getNodeLabel())
              Append topic element as a child of the root element

Export created elements into a XTM document
```

Figure 3.20: Algorithm to create RTH

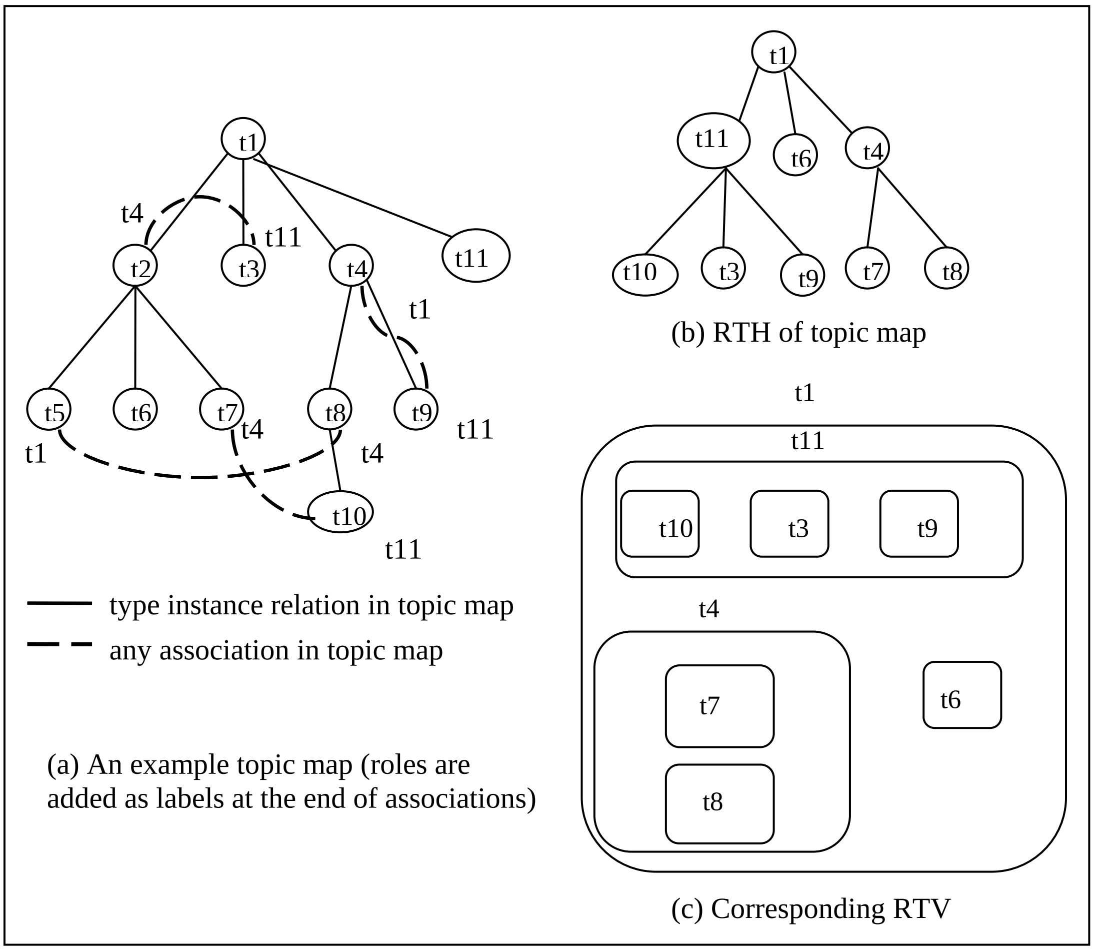

t1
t4
t11
t2
t3
t4
t11
t1
t5
t6
t7
t4
t8
t9
t11
t1
t4
t10
t11
type instance relation in topic map
any association in topic map
(a) An example topic map (roles are added as labels at the end of associations)
t1
t11
t6
t4
t10
t3
t9
t7
t8
(b) RTH of topic map
t1
t11
t10
t3
t9
t4
t7
t6
t8
(c) Corresponding RTV

Figure 3.21: Example of RTH and RTV

3.4.4 Occurrence Type Hierarchy (OTH) and Occurrence Type View (OTV)

Occurrences point the resources about the subject that a topic represents. An occurrence can be instance of a topic. For instance, topic ‘Jane’ has an occurrence that is reference to an image file. Furthermore, this occurrence has a type of “image” where “image” can be a topic or not. Type of an occurrence is specified by <instanceOf> child element.

An occurrence type hierarchy contains occurrence types and their ancestors in the topics of which occurrences exist. These topics becomes the children of their occurrences types if we look at the example above, occurrence type hierarchy contains the topics “image” and “Jane”. “Jane” becomes the child of the topic “image”. Assume that topic “image” has a type “media”. Then the topic “media” has also included in hierarchy as the type of the topic “image”. All topics are represented by blobs.

Algorithm of creation of OTH is given in Figure 3.22.

```
createOccurrenceTypeHierarchy(occurrNodesTable, topicsTable)

Input: all topics in the topic map and occurrence nodes found in TTH
Output: OTH
Algorithm:

Create a root element with the label “occurrenceTypeHierarchy”

For each node onode in occurrNodesTable
  Add onode into nodesT
  Set actors as rnode.getCustomizedData(“actors”)

  For each node actor in the actors
       If the actor topic is not created before create a “topic” element
       If no “xlink:href”attribute equals to onode.getNodeLabel() exists in descendant
       “topicRef” element
              Create a “instanceOf” child for the “topic” element
              Create a “topicRef” element as child of the “instanceOf” element
              topicRef.setAttribute("xlink:href","#"+onode.getNodeLabel())

       Append topic element as a child of the root element

       parent = onode.getCustomizedData(“parent”)
         While parent != null
              Add parent node into nodesT
              Parent = parent.getCustomizedData(“parent”)

  For each node node in nodesT
       parent = node.getCustomizedData(“parent”)

       If element for this topic is not created before create a “topic” element
       If parent != null and no “xlink:href”attribute equals to parent.getNodeLabel() exists in
       descendant “topicRef” element
              Create a “instanceOf” child for the “topic” element
              Create a “topicRef” element as child of the “instanceOf” element
              topicRef.setAttribute("xlink:href","#"+parent.getNodeLabel())

       Append topic element as a child of the root element

  Export created elements into an XTM document
```

Figure 3.22: Algorithm to create OTH

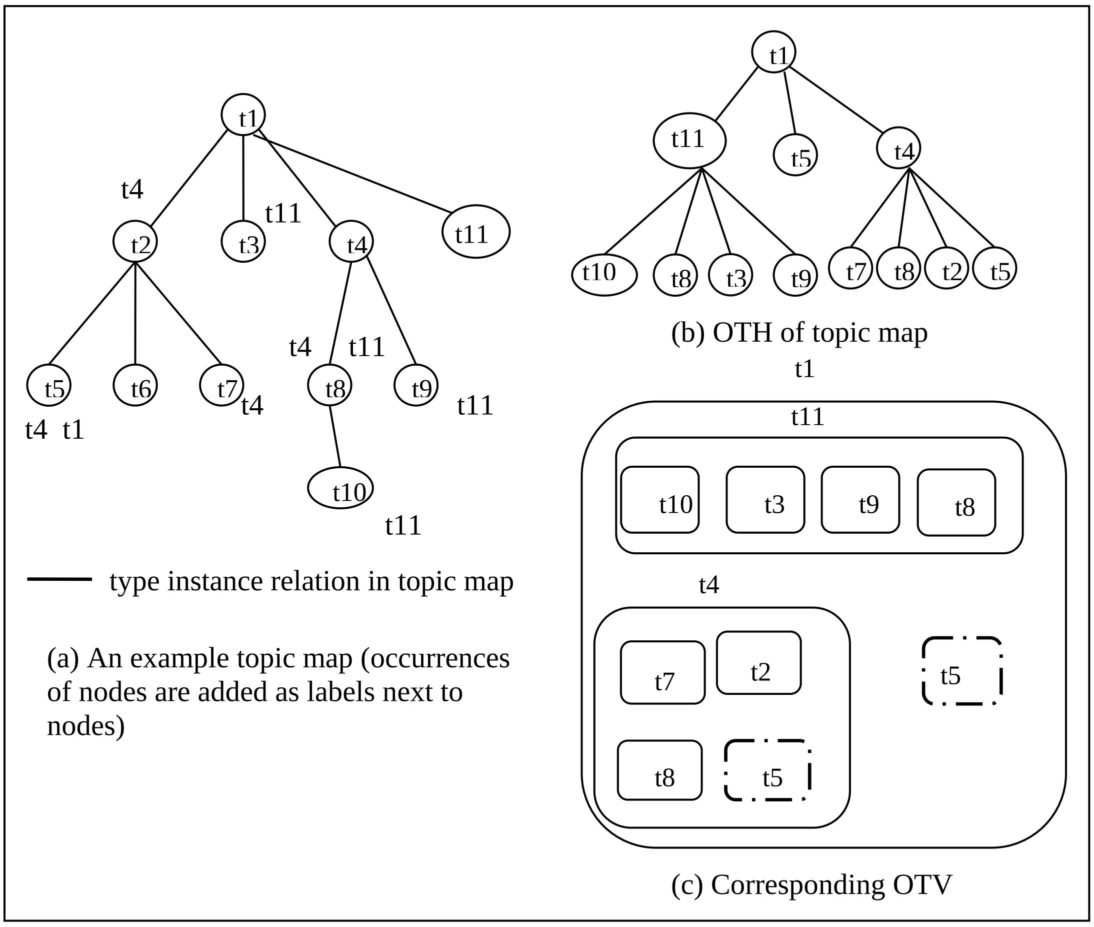


Figure 3.23: Example of OTH and OTV

The Figure 3.23 shows an example of OTH and OTV for a sample topic map. Occurrence types for owner topics are written close to topic blobs in TTH. As shown in the figure a topic may have multiple occurrences, as a result it may be shown in multiple blobs in OTV.

CHAPTER 4

PROGRAMMING LIBRARIES USED IN IMPLEMENTATION

We used some existing tools in our implementation that do visualization of graphs. We modified and adapted them to visualize topic maps using the technique that we have present. We used three existing tool: Piccolo 1.0, Protégé 2.1.1 and Shrimp 2.1.0. Piccolo (Piccolo) is a structural 2D graphics framework for building zoomable user interfaces developed at University of Maryland; Protégé (Protégé) is a tool to create and modify ontologies; Shrimp (Shrimp) is an application to visualize and explore information spaces developed by thechisel group.

4.1 PICCOLO

Piccolo is a tool to visualize 2D structural graphics. It is developed at Human-Computer Interaction Lab in University of Maryland. It supports hierarchical scene graph model, that is, it maintains a hierarchical structure of objects and cameras. This allows grouping, orienting and manipulating objects easily. Piccolo is a layer built on lower level graphics API, and allows application developers to build their animated graphical application without dealing with low-level details. It already supports hierarchies (transformation, transparency, etc.), animation, event handling and dispatch, cameras, layers, views etc. It provides efficient repainting of the screen, bounds management, picking (determining which visual object the mouse is over), animation, and layout. There are currently three versions Piccolo: Piccolo.NET, PocketPiccolo.NET (for the .NET Compact Framework) and Piccolo.Java. Piccolo is a free and open source application.

Objects on a Piccolo canvas are nodes (PNode), even texts. Application developers can define new nodes on top of those defined in Piccolo. All piccolo interfaces are placed in PCanvas so that they can be viewed and be interacted.

There are four main classes, which define the Piccolo's core (Piccolo):

- PNode (anything that is visible and gets events)
- PCamera (a node that looks at other layer nodes, and applies a view transform)
- PLayer (a node that can be looked at by a camera)
- PRoot (the root of the Piccolo display tree)
- PCanvas (the host component that lets PNodes exist in a Java Swing or .NET Windows application. Each PCanvas is associated with a PCamera. But all cameras are not necessarily directly associated with a PCanvas, internal cameras for example are not.)

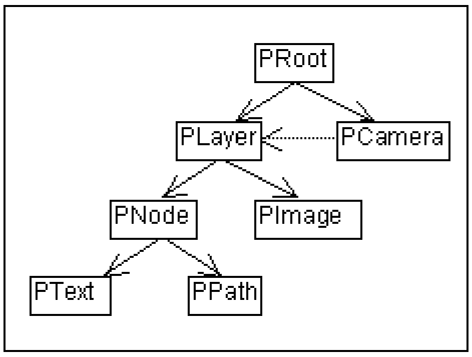


Figure 4.1: Piccolo Runtime Structure (Piccolo)

Figure 4.1 shows runtime structure of Piccolo. Several real-world applications have been built on Piccolo. A screenshot of one of the applications (Creole) is shown in Figure 4.2.

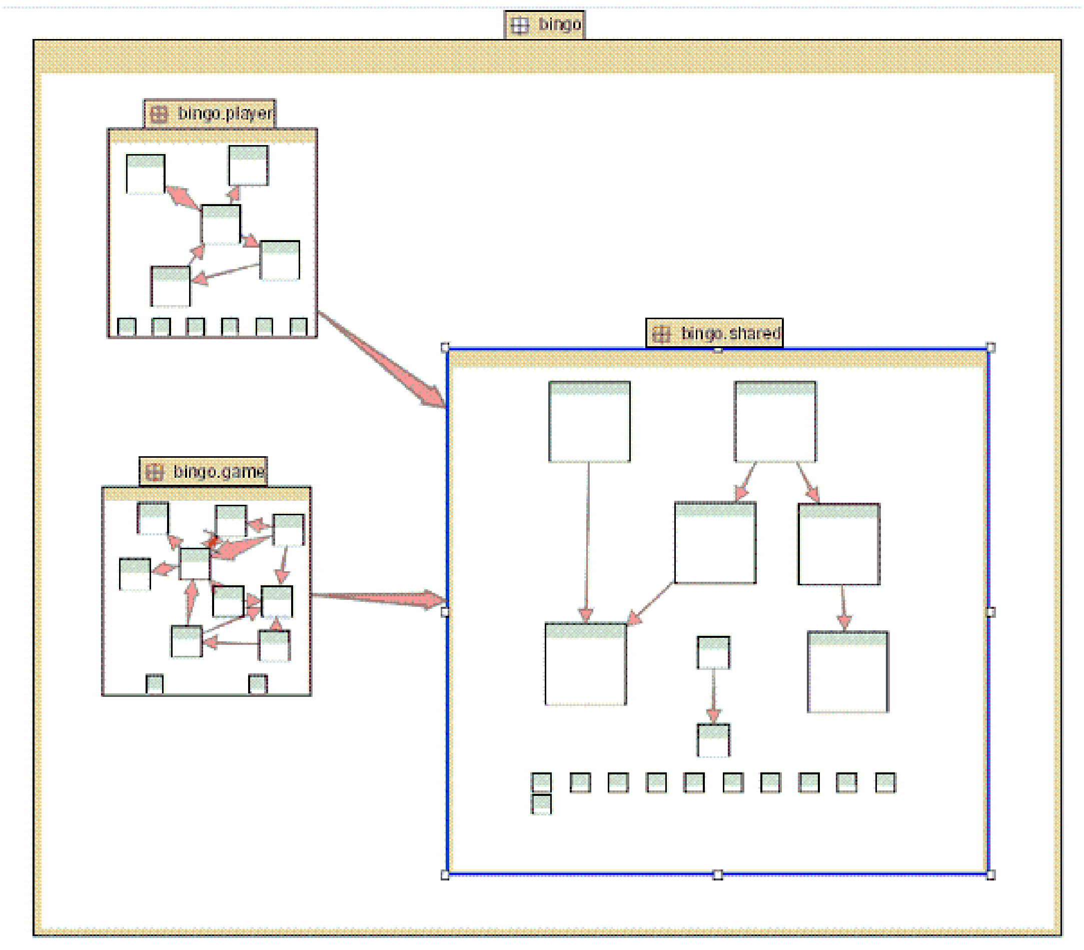
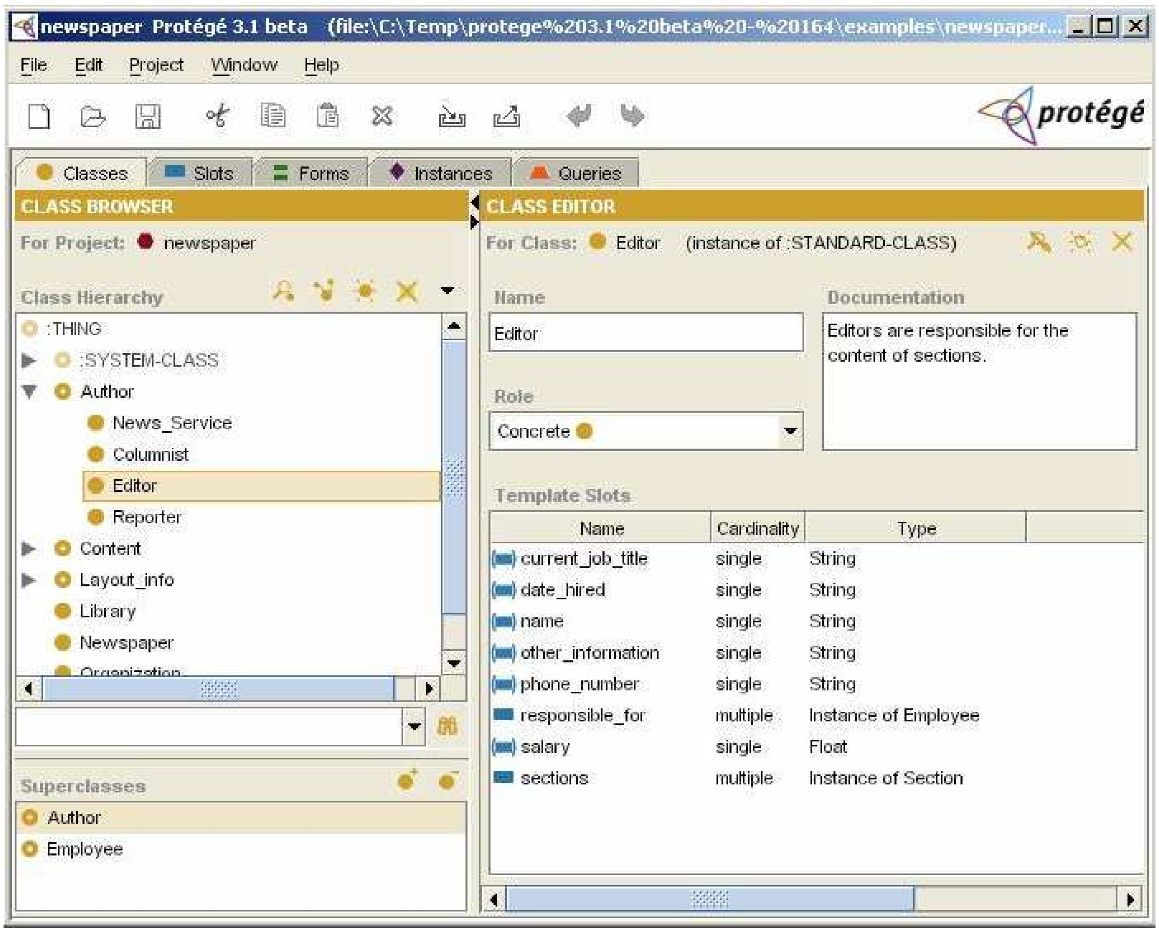


Figure 4.2: A screenshot of Creole Application (Creole)

4.2 PROTÉGÉ

Protégé is an extensible, platform-independent ontology editor and knowledge base framework. It is developed at Stanford Medical Informatics at Stanford University. Protégé allow user to define classes, class hierarchies, relationship between classes, and properties of those relationships. Instance of classes can be defined as well. A screenshot of the tool is shown in Figure 4.3.



newspaper Protégé 3.1 beta (file:\C:\Temp\protege%203.1%20beta%20-%20164\examples\newspaper...

For Project: newspaper

:THING
:SYSTEM-CLASS
Author
News_Service
Columnist
Editor
Reporter
Content
Layout_info
Library
Newspaper
Organization

Superclasses: Author, Employee

For Class: Editor (instance of :STANDARD-CLASS)

Name: Editor

Documentation: Editors are responsible for the content of sections.

Role: Concrete

Name	Cardinality	Type
current_job_title	single	String
date_hired	single	String
name	single	String
other_information	single	String
phone_number	single	String
responsible_for	multiple	Instance of Employee
salary	single	Float
sections	multiple	Instance of Section

Figure 4.3: A screenshot from the Protégé framework

Users can create and modify own ontologies in any format in the Protégé. They can export ontologies to different formats. Protégé displays all entities in different tabs (classes, slots, instances etc). User is allowed to modify the data while navigating ontologies.

Protégé is also supports to query ontologies. User can create, run and save queries that select instances from the project base.

Developers can write plug-ins and add to Protégé. A plug-in is an extension to Protégé. Protégé displays ontology data spread into tabs using text fields. It uses a plug-in to support 2D visualization of ontologies. Jambalaya is a plug-in in the Protégé framework that supports visualization of ontologies using Shrimp. We used Shrimp project in our thesis.

4.3 SHRIMP

Shrimp is an application to visualize and explore software architecture and any other information space. It visualizes "graph based" data formats such as GXL, RSF, XML, XMI and also Protégé knowledge-bases. It is developed by the CHISEL research group within the University of Victoria's Department of Computer Science. Shrimp uses Piccolo as low level graph API.

Shrimp provides a hierarchical graph view as shown in Figure 4.4.

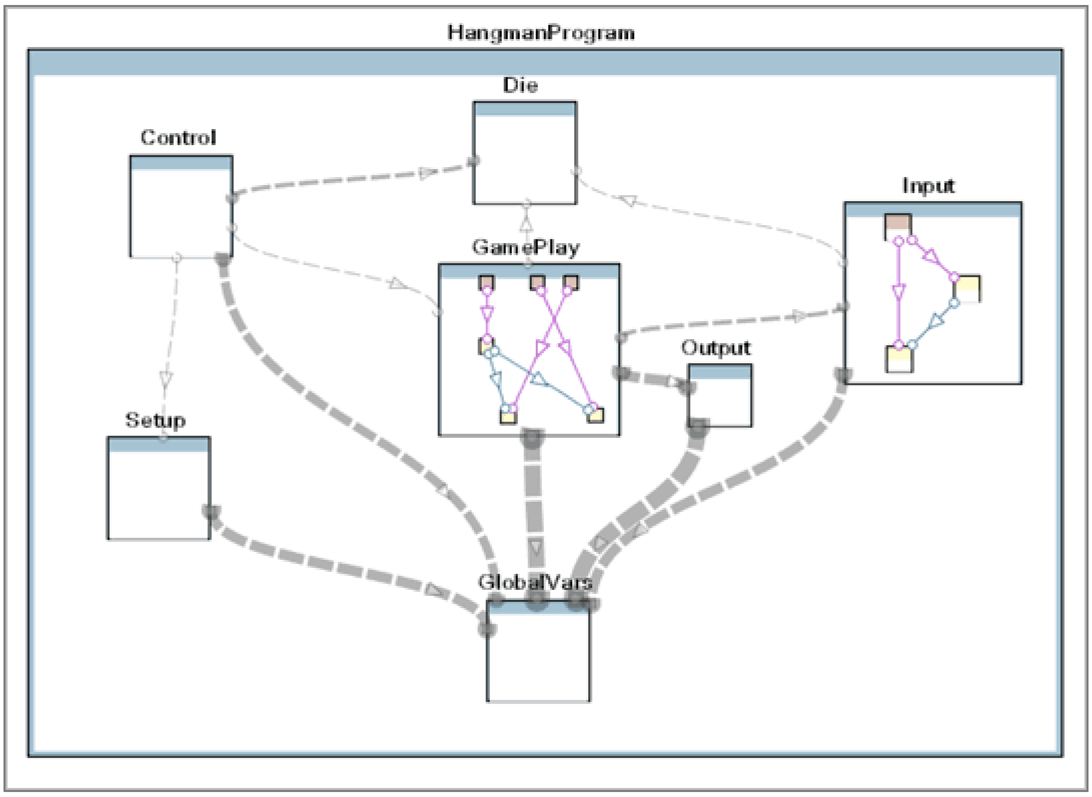


Figure 4.4: A screenshot from Shrimp project (Shrimp).

A Shrimp view consists of nodes and arcs. Shrimp provides several operations on nodes (select, open, close, zoom, magnify, filter etc) and arcs (go source, go destination, filter etc). It creates a hierarchy among nodes based on a relationship between them. User can select the relationship to create hierarchy and also can change the relationship later. Some of the operations that Shrimp provides are:

- On nodes
    - Select/Show children

    - Show all descendants
    - Select siblings
    - Close
    - Search
    - Filter
  - On arcs
    - Go source
    - Go destination
    - Search
    - Filter

Shrimp can create different layouts of the graph: Grid Layout, Radial Layout, Spring Layout, Tree Layout and TreeMap Layout. It also creates different views of the display scene. It allows user to change the shapes and colors of the nodes and arcs and size of the nodes. Further information is at (Shrimp).

CHAPTER 5

IMPLEMENTATION

Data model of our application was explained in the section 3. As we mention there, we represent a topic map with a multilevel higraph. As a summary, there are four different views (TTV, ATV, RTV, and OTV) for four different type hierarchies (TTH, ATH, RTH, and OTH).

Views are consists of blobs and may be arcs between blobs. Blob color, blob size, blob shape, arc type, arc thickness and arc style is used to indicate some features of the entities to the user. Usually, blob color distinguishes whether the entity represented by the blob is a base entity or type entity, that is, whether it is an object or type. Blob shape is also used as the same feature like blob color. Blob size indicates the number of descendant blobs if it is not a blob at the bottom level. Arc color is used to differentiate the association types. Arc style corresponds to blob color that it distinguishes whether the edge represented by the arc is a base edge or a metaedge. Thickness of an arc indicates the number of descendant arcs if it is an arc representing a metaedge. These decisions are applied into all views. From now on, we will use metaarc, metaedge and metaassociation interchangeably as well as baseassociations and baseedges .We can use metaarcpart instead of saying metaedgepart, too.

In TTV, topics and association between those topics are displayed. Topics (both type topics and base topics) are represented by blobs, and associations are represented by arcs. Blobs have labels showing the names of the topics.

In ATV, associations and type topics are displayed. Both represented by blobs, while associations are the blobs at the bottom level of graph.

In RTV, roles, player topics of roles and the type topics of roles are displayed. All represented by blobs. Player topics are represented by the base blobs in the view.

Occurrences, the type topics of the occurrences and the topics having those occurrences are displayed in OTV. Base blobs represent topics having the occurrences.

In the following sections we will explain implementation of our work to visualize TM. Visual operations in our work will be given in details and our extensions to tools we used and modifications will be explained.

5.1 IMPLEMENTATION AS AN EXTENSION OF SHRIMP

We developed our tool in JAVA using Eclipse 3.0 IDE environment. Essentially, we utilized the Shrimp project whereas it uses Piccolo and runs with some packages of Protégé project.

Shrimp implemented the hierarchical layout of blobs and the representation of binary edges for any information organized in nodes and arcs. It also provides main operations on blobs with a little on arcs that we have planned to implement.

One of the main tasks in our project was to convert data in a topic map into nodes and arcs. Shrimp was able to read documents in XML, Shrimp project or Protégé project format and create nodes, arcs, and some hierarchies between nodes. We wrote new java classes and methods to read XTM files and extract the topics, associations, roles, occurrences and type hierarchies of them.

First, topics and associations are extracted from the file and topics are saved in special data structure defined in Shrimp whereas we created a new structure to store associations. Shrimp does not support hyperedges. We implemented new association structure using binary edge and node structures in the Shrimp. An association in our implementation consists of a node (we call it hypernode) in its middle point and binary edges. Hypernodes are typical nodes except they are displayed in small sizes. Number of binary edges in an association is equal to the number of member nodes. One end of each edge is connected to a member node while other edges are connected to the hypernode corresponding to association. As a result, a hyperedge structure is produced as shown in the following Figure 5.1 Each association in a topic map data is stored in the hyperedge structure even if they are binary or not except the metaarcparts. Because they are always binary edges between children and parent and keeping them as binary edge provide us efficiency in the application since hyperedge structure brings more coding during the manipulations of associations.

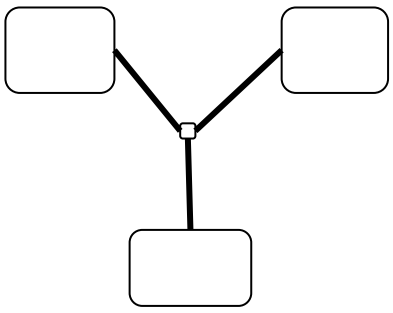


Figure 5.1: Example representation of a hyperedge using binary edges

All parts in the Shrimp that deals with arcs are modified in order to adapt them into hyperedges. In addition, layout algorithm of nodes is modified and a different layout algorithm (HyperNodesLayout.java) is written for hypernodes since they are displayed at the center of hyperedges. A hypernode is located in the same distance to center points of the member blobs. If it falls in the area of a member blob then it is moved out the containing blob. Existing operations on nodes in Shrimp are also modified so that hypernodes are not affected by those operations. New actions for hypernodes are written such that visibility of hypernodes is changed by the visibility of edges.

When topics are retrieved from the document, type hierarchy among the topics is calculated. Thus, topic, association between topics and TTH are created in the application. Three separate methods are written to calculate and create ATH, OTH and

RTH. Those are calculated after all topics and associations in a document are read. In those methods, type hierarchies are calculated and the new hierarchy is saved into different documents in topic map format. Actually, a back operation is done here such that we created some type hierarchies from a topic map and store the hierarchies in TM documents.

In ATH, a new topic is created for each association and stored in the ATH document. Type attribute of these topics are set to the type topics of the associations. Topics that are type of associations and their ancestors are copied from topic type hierarchy preserving parent-child relationships and stored in the ATH document. So a new topic map consisting of only topics created.

In RTH, new topics are created for roles and stored in the RTH document. Player topics of the roles are copied to the document with the modifications in type attributes. Their types are set to the roles which they play. Notice that a topic may play several roles and so it may have multiple types. Topics that are type of role topics and their ancestors are copied from topic type hierarchy preserving parent-child relationships and stored in the RTH document. Thus a new topic map consisting of only topics created for RTV.

Topics that are type of occurrences and their ancestors in topic type hierarchy are copied into a new topic map document. Topics having those occurrences are also copied to the document by setting type attributes as types of occurrences they have. As in RTH, topics may have multiple types here, hence multiple occurrences they may have.

After the topic map document is parsed and topics and associations are extracted, metaassociations and metaarcparts are created. To do this, another class (MetaAssociation3.java) is written. It uses the topics and the associations extracted in TMPersistentStorageBean.java class and constructs metaassociations so hypernodes, metaarcparts and the hierarchy among themselves. The algorithm of construction of metaassociations, hypernodes and metaarcparts is given in the section 3.4.1. Arc thickness based on number of descendant arcs is also calculated and this information is attached to each arc here. Created arcs and nodes are stored in the common arcs and nodes tables created during the parsing of topic map document. Consequently, TTH is constructed with all topics and associations in the topic map plus hypernodes, metaasociations, and metaarcparts created in the application.

User first sees the TTV of the current topic map in the application. When he/she requests to see the other views, for example ATV of the topic map, application loads the created ATH document, reads it as a usual TM document and opens the view. Following figures shows screenshots of the school.xtm topic map's TTV, ATV, RTV and OTV.

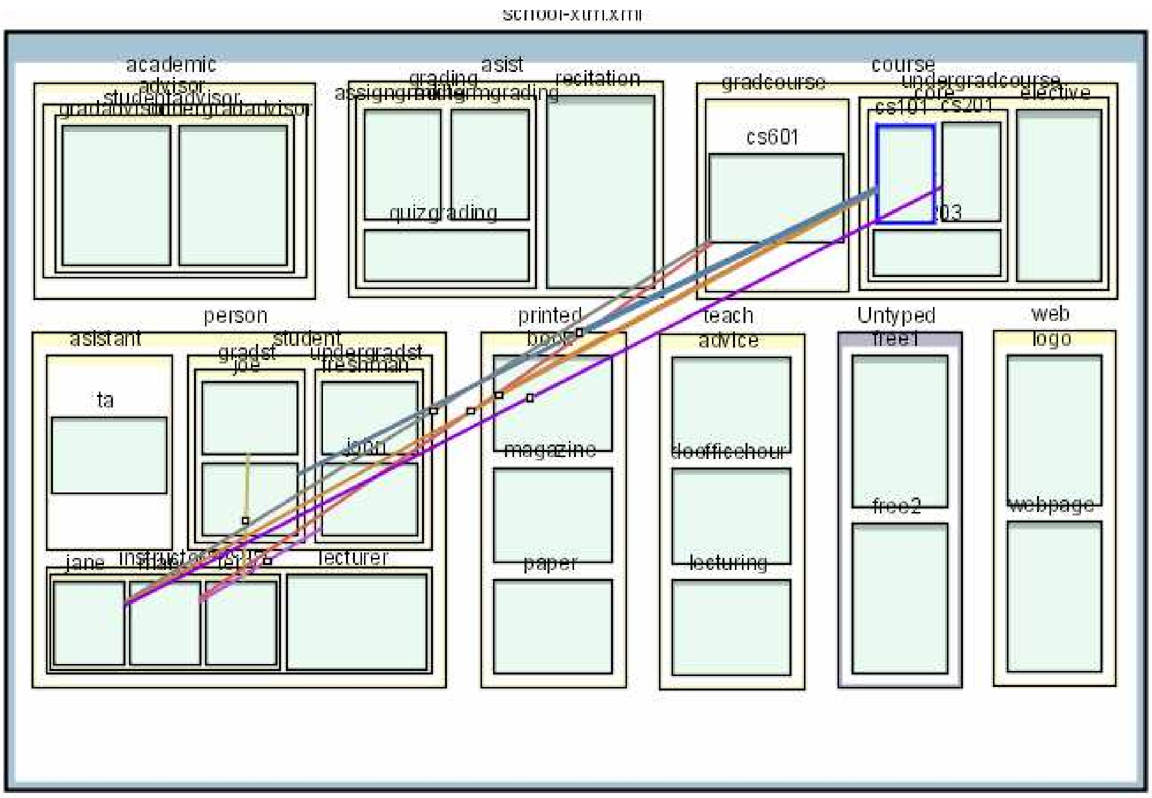


Figure 5.2: TTV of school.xtm

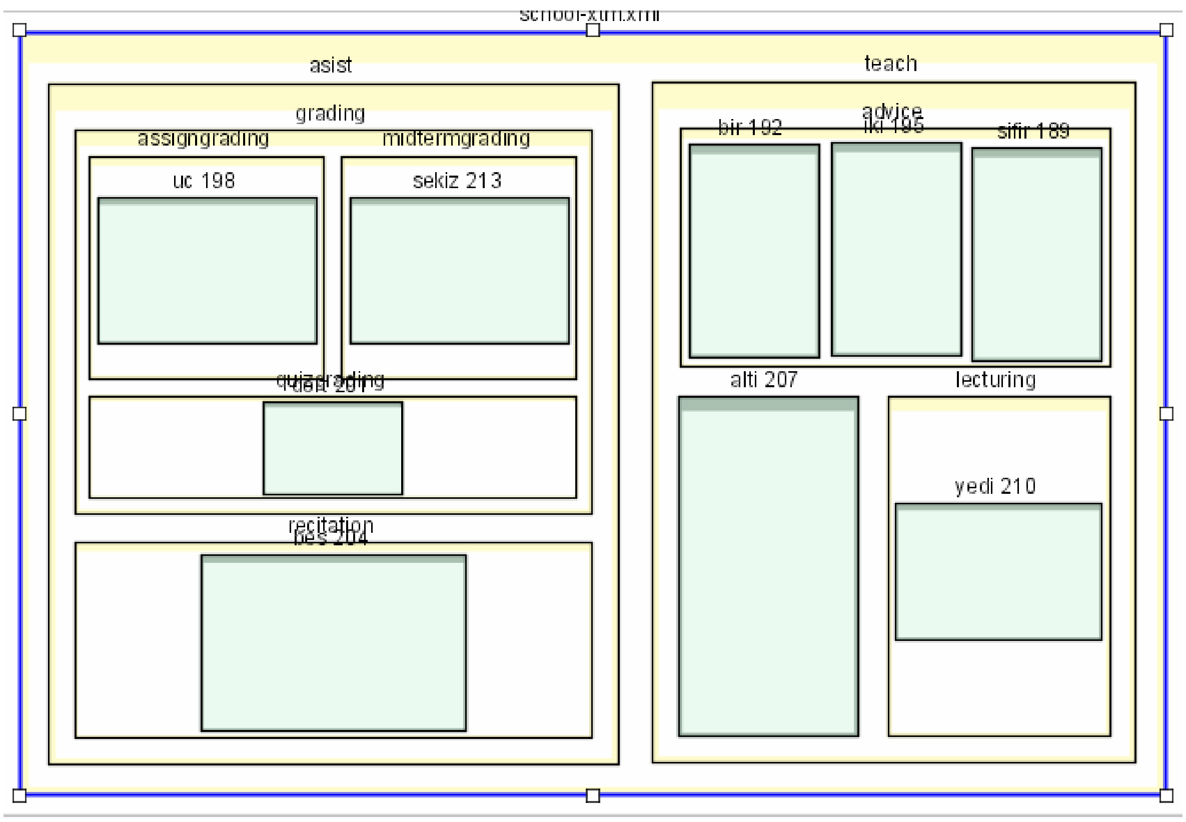


Figure 5.3: ATV of school.xtm

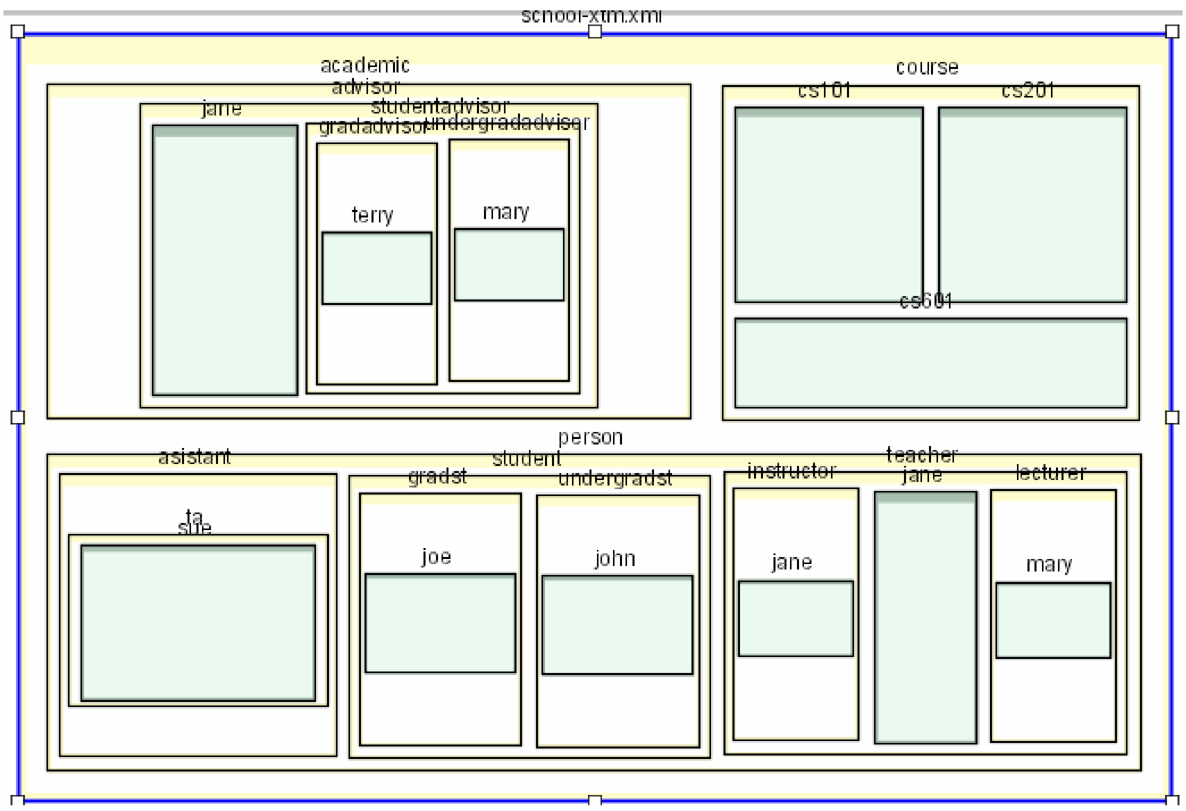


Figure 5.4: RTV of school.xtm

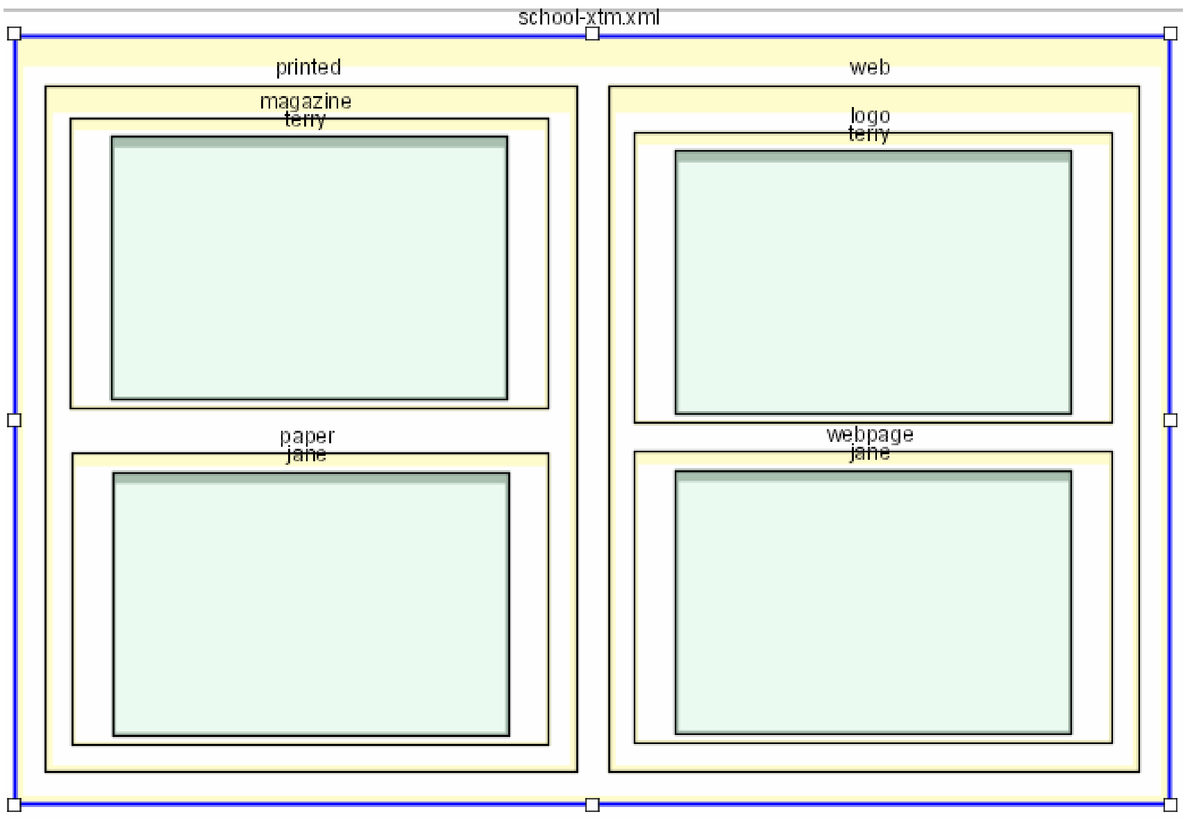


Figure 5.5: OTV of school.xtm

5.2 OPERATIONS

Hierarchical views of TM allows user to explore and navigate a topic map in the form of a multilevel higraph using abstract levels of hierarchies. At the start user is presented with top most view that is only the root node in the hierarchy. User can explore the topic map using a set of operations defined below. These operations can be grouped into three sets: operations on blobs, operations on edges, and operations on the graph.

Operations are derived from the information visualization techniques which follows the principle “Overview first, zoom and filter, then details on demand” (Shneiderman, 1996). Basic operations focusing, zooming and filtering are listed in Table 5.1.

1. Focusing Operators

   Select/deselect: This operator changes focus in the TM. Individual or groups of selected blobs, metaedges can be brought to focus for further manipulations.

2. Zooming Operators

   Zoom/un-zoom: Displays or hides the contents of a blob, metaedge, or a level. The contents that is displayed will be the lower level objects in the selected item.

3. Filtering Operators

   a. *Hide/show*: This operator selectively hides/shows blobs or metaedges. It is used for filtering out parts of a topic map that is not of interest to the user.

   b. *Filter/un-filter*: This operator is used to filter out details that clutter the view.

   c. *Group/un-group*: This operator allows user to group a set of blobs and metaedges into a blob. A new set of metaedges between the new blob and existing blobs has to be computed. Grouping hides content of the new blob.

Operator	*Blob*	*Meta edge*	*Graph*
Select/de-select	Yes	Yes	Yes (multi selection)
Zoom/un-zoom	Yes	Yes	Yes (go up/down 1 level)
Hide/Show	Yes	Yes	Yes
Filter/un-filter	Yes	Yes	Yes (apply filter/un-filter to all objects)
Group/un-group	Yes	Yes	Yes (levels can be coalesced)

Table 5.1: Operator applicability table

5.2.1 Operations on Blobs

5.2.1.1 Primitive Operations

select: User by clicking on the corresponding blob brings the topic to the focus, thus further operations can be applied. Selected blob is highlighted. Normally the previously selected blob if exists will be deselected. Associations connected to the selected blob are also highlighted. This operation was implemented by Shrimp. Selection of “asist” topic in school.xtm is shown in Figure 5.6.

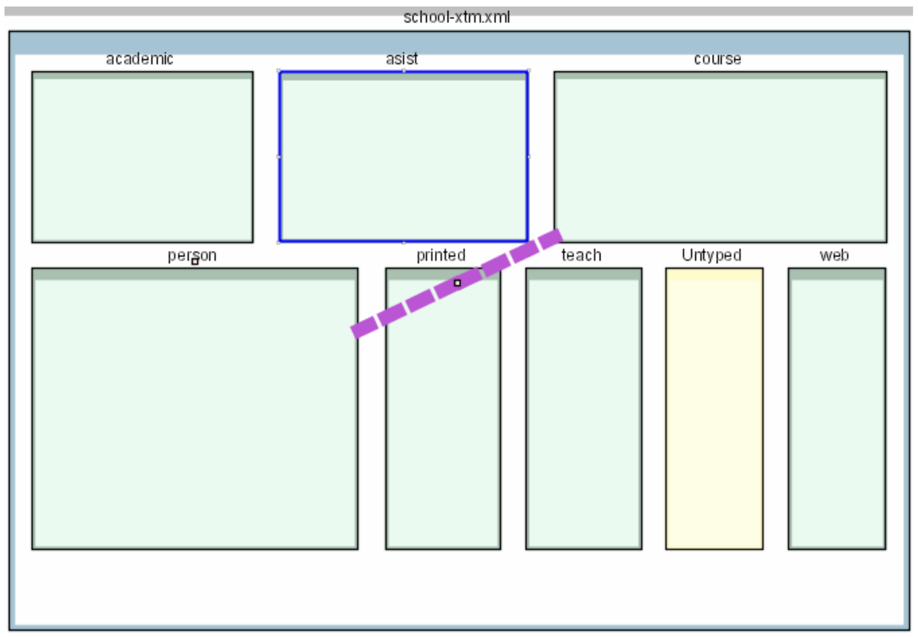


Figure 5.6: select operation

setVisible(true/false):

true → node (we use node and blob interchangeably, afterward) is displayed on the screen if the parent node is in display, that is, if the parent node is visible. If node is already visible or parent node is not visible then nothing is done. If node is displayed then;

i. metaarcparts between the node and the parent that are related to any visible metaassociations are displayed.

ii. base/metaassociations between the node and its visible siblings are displayed.

pseudo code:

```
node.setVisible(true) :
      if node.getParent().isVisible()==false
         return;
      else
         display the node on the screen
         associations = node.getAllAssociations()

         for each association assoc in the associations
              if assoc is an instance of base/metaassociation
                     if other member of the assoc is a sibling of the node and
         the sibling is visible
                            assoc.setVisible(true)
                 if assoc is an instance of metaarcpart
                            if other member of the assoc is parent of the node
                    and related association to this assoc is visible
                            assoc.setVisible(true)
```

false → if node is in display then it is hidden. If node is already invisible then nothing is done. If node is hidden then;

i. metaarcparts between the node and the parent that are visible are hidden.

ii. base/metaassociations connected to the node that are visible are hidden.

pseudo code:

```
node.setVisible(false) :
      if node.isVisible()==false
         return;
      else
         hide the node on the screen
         associations = node.getAllAssociations()

         for each association assoc in the associations
              if assoc.isVisible()==true
                     assoc.setVisible(false)
```

In order to keep primitive operations simpler we have defined setVisible operation in the context of parent node. Calling setVisible() method for any node only affects the content and display of parent of the node. One can define a setVisible() operation that displays the node as well as all associations connected to node and the other nodes that are exists in anywhere of the tm. Sample view of the setVisible operation can be shown in the example figure of the open operation where all children of the selected node is set visible.

Shrimp had implemented setVisible() operation for nodes but we modified the method AbstractDisplayBean.setDiplayObjectVisible(ShrimpDisplayObject sdo, boolean visible, boolean assignDefaultPosition). In the original method node was set to visible/invisible and setVisible() operation for the arcs connected to that nodes was called with the same visibility. In our implementation, after setting the nodes visibility, hypernodes of the metaassociations related to that node are set to the same visibility. In addition, we added some codes to prevent nodes to open descendents association of a visible metaassociation. Because associations will be visible by double clicking on parent association. If any of ancestor association an association is visible, method do not set this association visible.

Following primitive operations were implemented by Shrimp.

isVisible(): For nodes that are type or topic nodes, it returns true if the node is displayed, else returns false.

getType(): Returns the type of a node.

getParent(): Returns the parent node of a node.

setParent(): Sets the parent node of a node.

getChildren(): Returns the children nodes of a node.

getAllAssociations(): Returns all associations connected to a node.

5.2.1.2 Complex Operations:

open: Displays the content of a node. The contents of a blob are the sets of lower level blobs (children), metaassociations among them and metaarparts between the node and the children.

pseudo code:

```
open(node):
      if node.isVisible()==false
         return;
      else
         children = node.getChildren()

         for each node c in the children
              c.setVisible(true)

         setVisibleHyperNodesPositions()
```

open() operation was implemented by Shrimp we modified it in order to recalculate positions of hypernodes. When a node is opened so children are set visible new association may be set visible also. Hypernodes of the new opened association must be placed in the view. A new method setVisibleHyperNodesPositions() is added to the AbstractDisplayBean.java to set positions of visible hypernodes. This method is called in the open(node) method. It finds the visible hypernodes and calls HyperNodesLayout algorithm to calculate new positions.

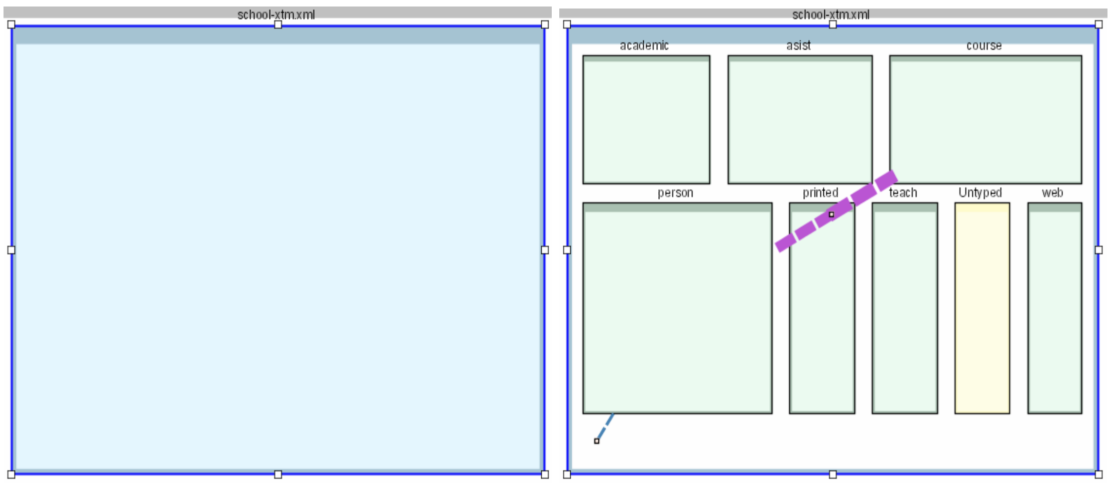


Figure 5.7: open operation

Figure 5.7 shows open operation applied to the root node of the school.xtm. All children nodes and associations among them are displayed.

Close: Hides the content of a node. The contents of a blob are the sets of lower level blobs (children), metaassociations among them and metaarparts between the node and the children. In Figure 5.7 we can see close operation on root node where second figure displays root node opened and first figure displays root node closed.

pseudo code:

```
close(node):
    if node.isVisible()==false
        return;
    else
        children = node.getChildren()

        for each node c in the children
            c.setVisible(false)
            for each hypernodes hn of the associations connected to the c
                hn.setVisible(false)

        for each hypernode hn of the associations connected to the node
            hn.setVisible(true)
```

We modified the close() operation implemented in the Shrimp. According to our definitions when a node is closed, that is children of the node is hidden, the associations connected to children will also hidden. If it is allowed, parent association of the hidden ones will be set visible. In order to implement these, we review all children associations and set them invisible, then find associations connected to this node and set them visible.

OpenAll: Displays the content of a node and the contents of all descendant nodes. A java adapter class was written in the Shrimp for this operation. Figure 5.8 displays openAll operation applied on root node of the school.xtm topic map.

pseudo code:

```
openAll(node):
    if node.isVisible()==false
        return;
    else
        children = node.getChildren()
```

```
for each node c in the children
    c.setVisible(true)
    openAll(c)
```

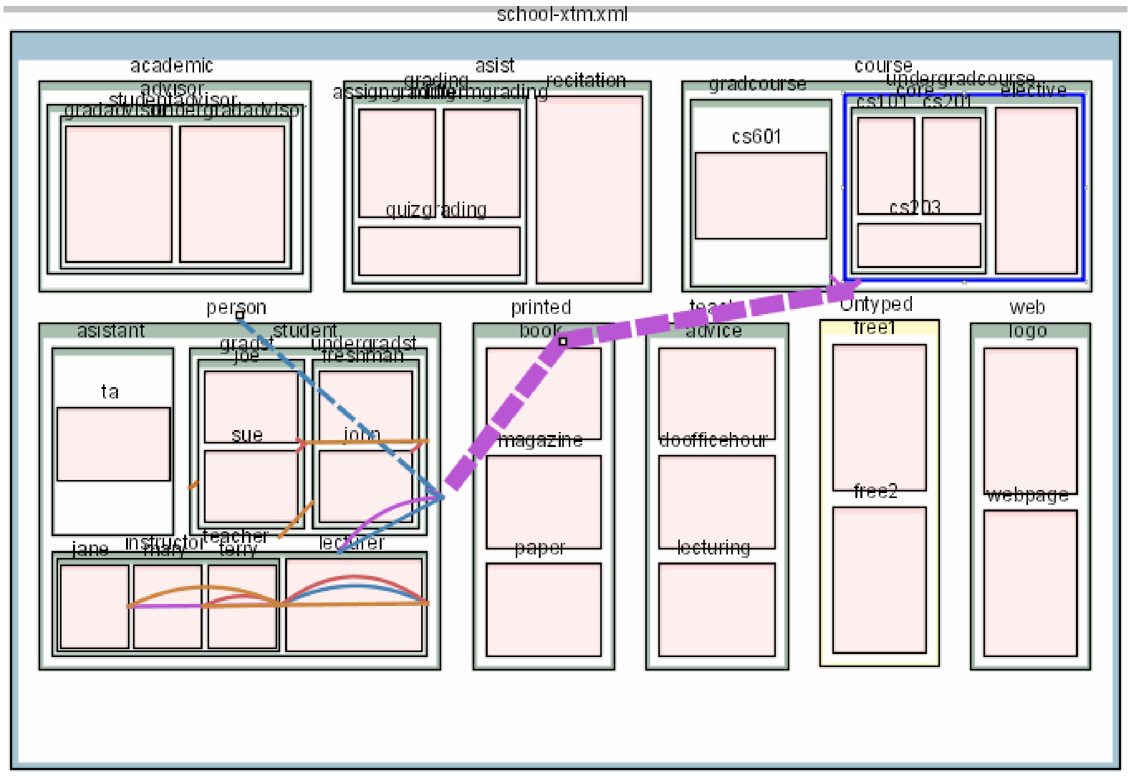


Figure 5.8: openAll operation

CloseAll: Hides the content of a node and the contents of all descendant nodes. CollapseAllDescendantsAdapter.java class was written to implement closeAll operation in the Shrimp. We did not modify it.

pseudo code:

```
closeAll(node):
    if node.isVisible()==false
        return;
    else
        children = node.getChildren()

        for each node c in the children
            closeAll(c)
            c.setVisible(true)
```

5.2.2 Operations on Associations

5.2.2.1 Primitive Operations:

select: User can select a base or meta association and bring it to the focus by clicking on the hypernode at the center of the association. Selected association is highlighted.

User can not select a metaarcpart although he/she can highlight a metaarcpart by moving the mouse over the arc. Operation list for any arc can be accessed by right clicking on the arc.

Shrimp did not supply to select an arc. We provide the selection of associations by selection of hypernodes of the associations. An example of select operation is shown in Figure 5.9.

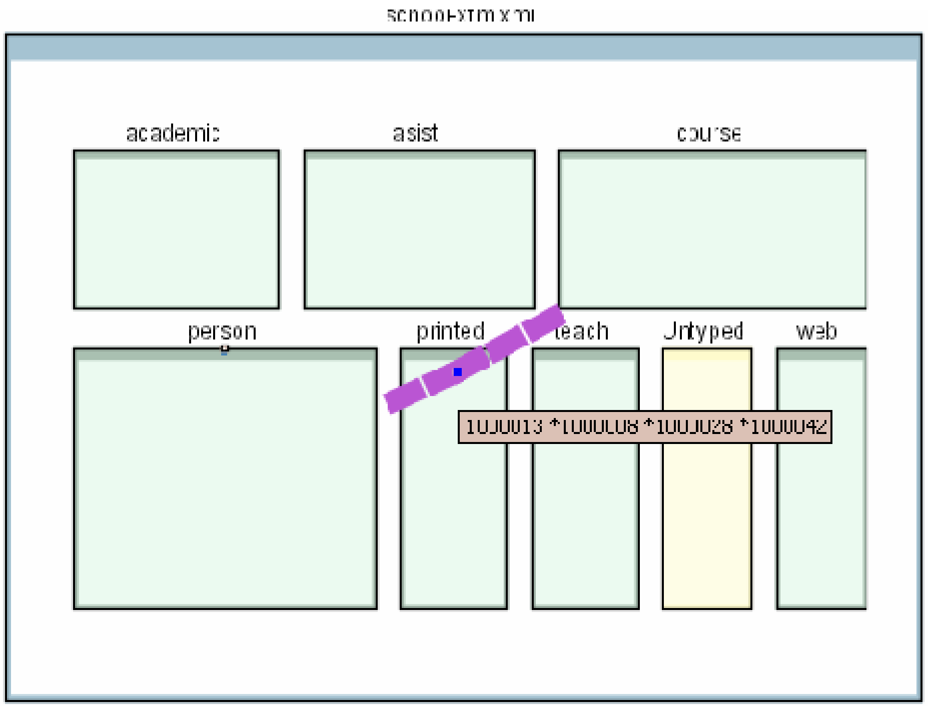


Figure 5.9: select operation on associations

setVisible(true/false): Sets the visibility of calling association. Original operation setVisible() in the Shrimp is modified to handle the visibility of metaarcparts too.

true → association is displayed if the member nodes is in display that is visible and the parent association if exists is not visible. If the association is already visible or parent association is visible then nothing is done. If the association is displayed then, metaarcparts related to this association are also displayed.

pseudo code:

```
association.setVisible(true):
        if association.isVisible()==true
            return;
        else
          display the association on the screen
            metaarcs = association.getMetaArcParts()

            for each association a in the metaarcs
                    a.setVisible(true)
```

false → Hide an association if it is in display. If the association is already invisible then nothing is done. If the association is hidden then, metaarcparts related to this association are also hidden.

pseudo code:

```
association.setVisible(false):
        if association.isVisible()==false
            return;
        else
          hide the association
            metaarcs = association.getMetaArcParts()

            for each association a in the metaarcs
                    a.setVisible(false)
```

isVisible(): It is called for an association and returns true if the association is displayed, else returns false.

getType(): Returns the type of an association.

getMetaArcParts(): Returns the metaarcparts related to an association if any exists.

getChildren(): Returns the children associations of an association if it is a metaasociation.

getParent(): Returns the parent association of an association if any exists.

getMembers(): Returns the member nodes of an association.

5.2.2.2 Complex Operations:

Complex operations are defined for metaassociations only, since there is no child association of any metaarcparts or baseassociations. Shrimp do not support any complex operation on associations. We implemented all the complex operations and embedded in the Shrimp.

expand: Sets an association invisible and the children of the association visible when a user double clicked on an association. Children associations can be visible if member nodes are in display. A new class MouseDoubleClickArcAdapter.java is written for this operation. It hides the association and shows the children associations. Hypernodes of the associations is placed in the view by calling the addhypernodestodisplay() method of the java class. This method is also implemented by us to place hypernodes in the display.

pseudo code:

```
expand(association):
        if association.isVisible()==false
           return;
        else if association.getType() == metaassociation
           association.setVisible(false)
           children = association.getChildren()

           for each association a in the children
                  a.setVisible(true)

          addhypernodestodisplay()
```

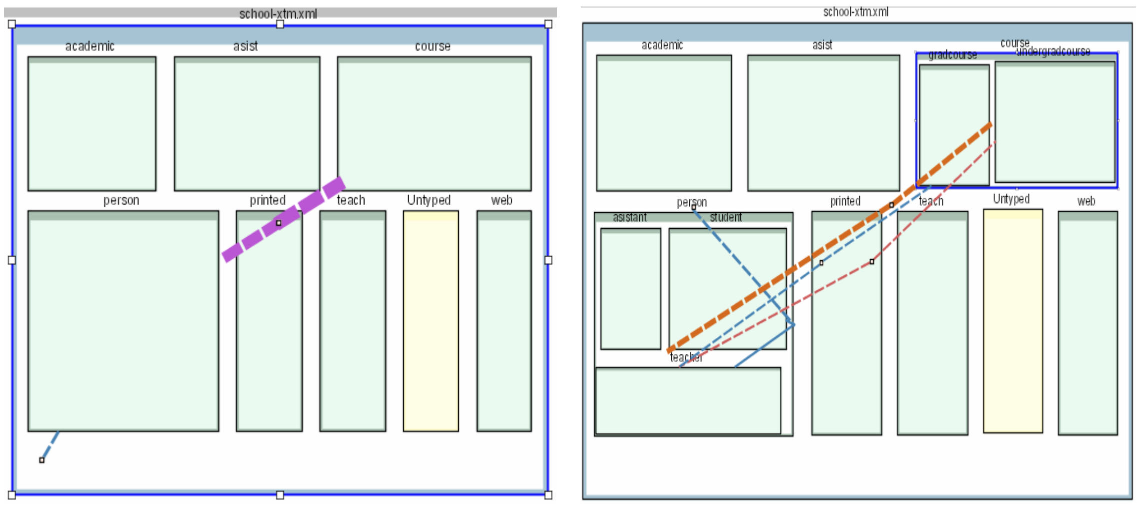


Figure 5.10: expand operation on associations

Expand operation on the hyperedge between person and course nodes is shown in Figure 5.10. This operation displays the children associations of the selected hyperedge and the children of member nodes.

collapse: Reverse of the expand operation. Sets an association visible and the descendants of the association invisible. This operation is not written as a separate method however implemented in the close(node) method. User cannot run the operation, it is only fired when a node is closed since the arc will be collapsed is invisible. Sample view of this operation can be shown in the Figure 5.10 whenever at least one of the member nodes is closed either person or course or both the association between them is collapsed.

pseudo code:

```
collapse(association):
    if association.isVisible()==true
        return;
    else
        children = association.getChildren()

        for each association a in the children
            a.setVisible(false)

        association.setVisible(true)
```

expandAll: Hides an association and displays all descendant base associations. If member nodes of the base associations are not visible, they are set to visible as well.

We wrote an adapter class ExpandAllBaseAssociationsAdapter.java to implement the operation. It first expands all member topics of the association. During this operation, it finds the descendant base associations. Then it sets the base associations visible and recalculate the layout of the hypernodes by calling the setVisibleHyperNodesPosition() method of the AbstractDisplayBean.java class.

pseudo code:

```
expandAll(association):

     if association.isVisible()==false
        return
     else
      memberNodes = association.getMembers()
      expandAll(memberNodes)
      openBaseAssociations()
     associations.setVisible(false)

expandAll(memberNodes)
      for each node n in the memberNodes
            openNodeRecursively(n)

openNodeRecursively(node)
      open(node)
      find the baseassociations related to that node
      arcsToOpen.add(baseassociaitons)

      for each child node c of the node
            openNodeRecursively(c)
openBaseAssociations()
      for each association a in arcsToOpen
            a.setVisible(true)

      setVisibleHyperNodesPosition()
```

Figure 5.11 shows expandAll operation applied on the association between person and course nodes.

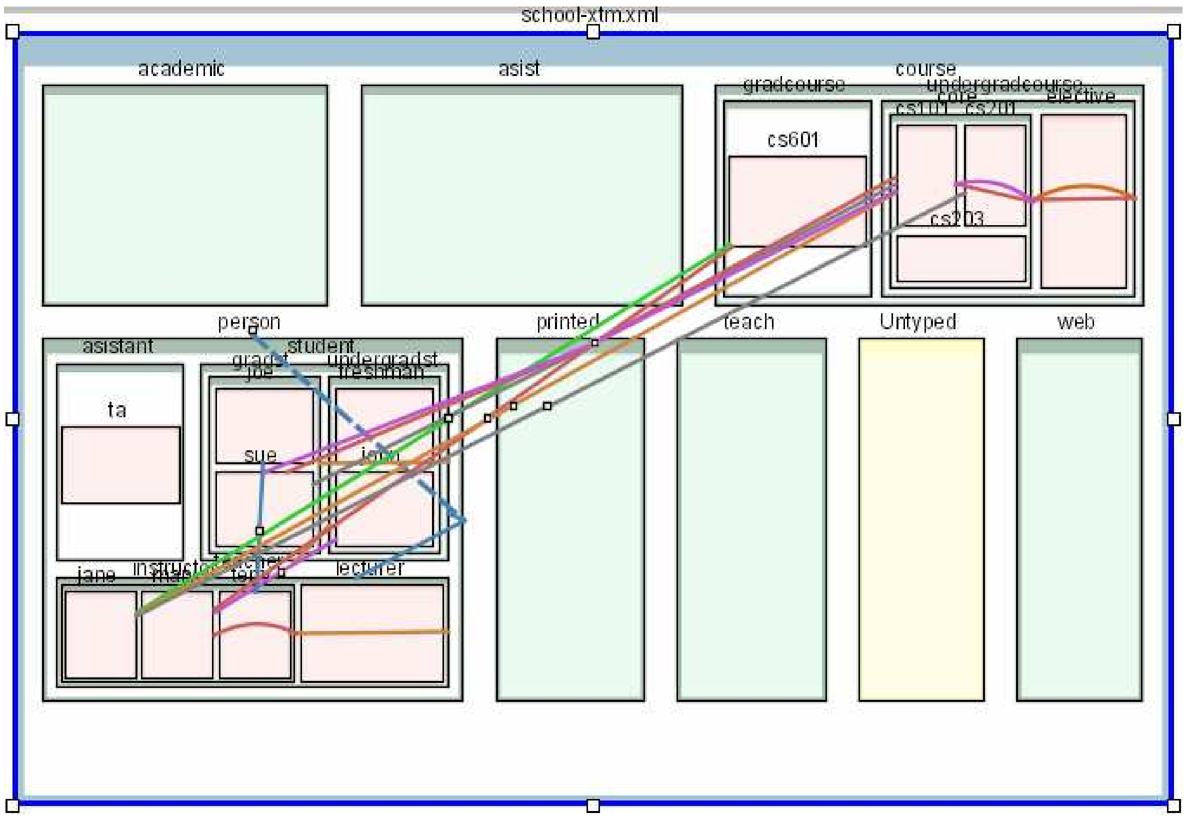


Figure 5.11: expandAll operation on associations

5.2.3 Operations on MultiLevelHigraph

5.2.3.1 Primitive Operations:

getVisibleNodes(): Returns the nodes in the current multilevelhigraph. Those are the nodes displayed on the screen.

getVisibleArcs(): Returns the arcs in the current multilevelhigraph. Those are the associations displayed on the screen.

5.2.3.2 Complex Operations:

Open/Close: Open/Close all blobs and associations at the same level of the graph. In effect, it displays the multilevel higraph representing one level lower/higher in TTH. The topic map itself is the lowest level. Figure 5.12 displays two graphs where an open operation is applied on first graph then second graph is obtained.

We implemented open() operation for graph. We wrote an adapter that select the visible descendent nodes and opens them. Thus a lower level graph is displayed.

We did not implemented the close() operation for graph yet.

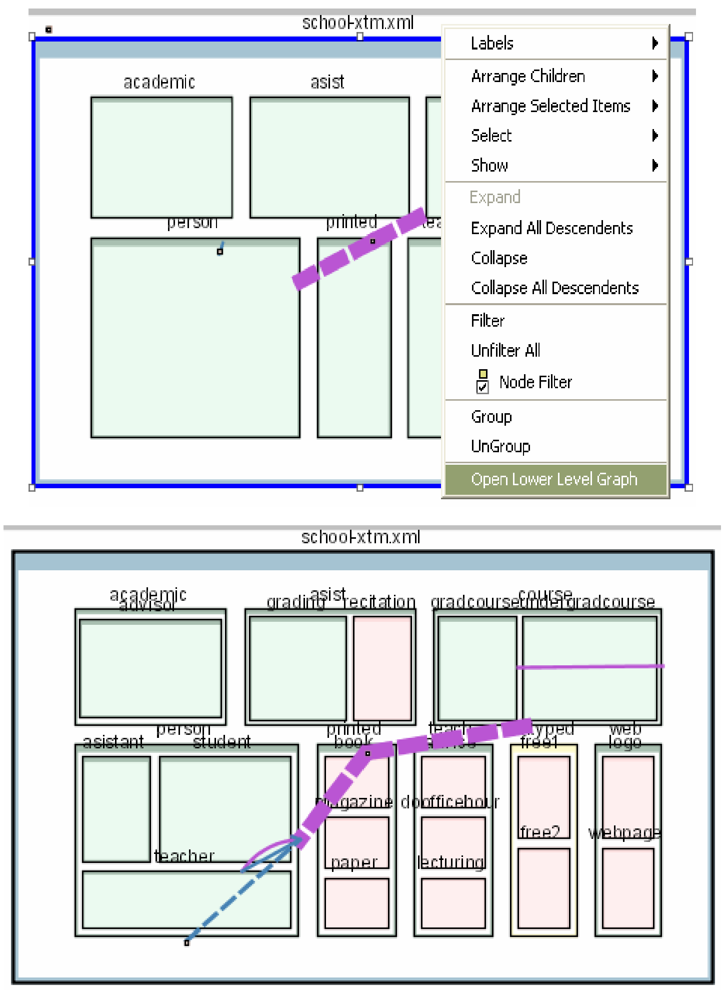


Figure 5.12: open operation on graph

Multi Selection: Allows selection of a set of blobs and associations, to be used in some other operation. Shrimp provides selection of the nodes having same type and allows user to select nodes by mouse click. Selection of three nodes is displayed in the following figure.

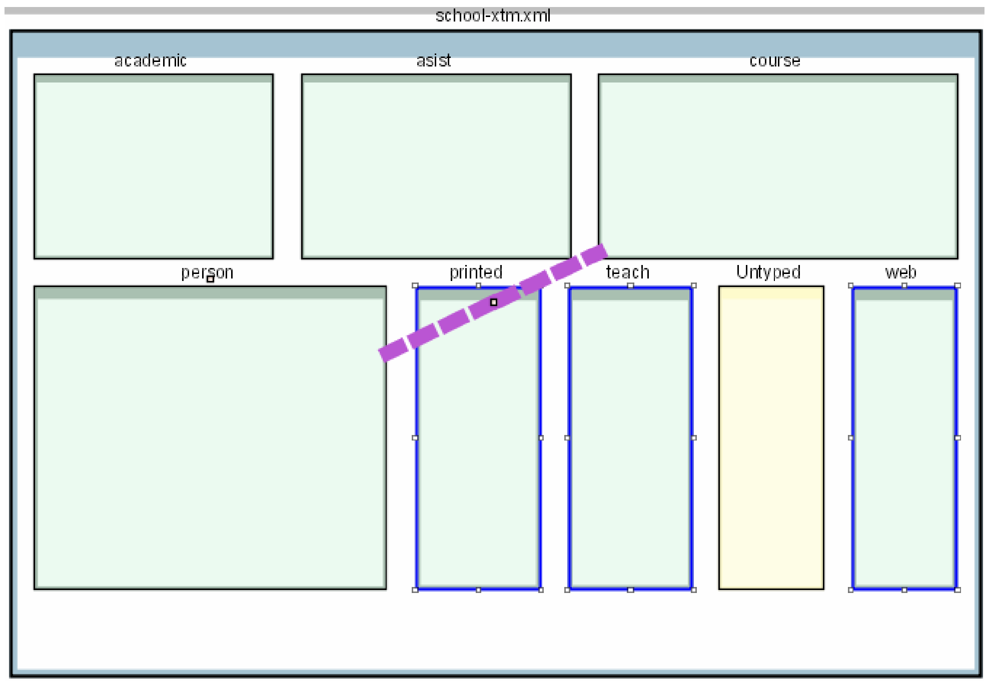


Figure 5.13: multiselection operation on graph

Open selection: It allows opening multiple blobs/meta edges.

```
pseudo code:
if multiple nodes selected:
open(selectedNodes):
        for each node n in the selectedNodes
             open(n)

 if multiple associations selected:
open(selectedAssociations):
        for each association a in the selectedAssociaitons
             expand(a)
```

Figure 5.14 represents open operation on the selected three nodes in the Figure 5.13.

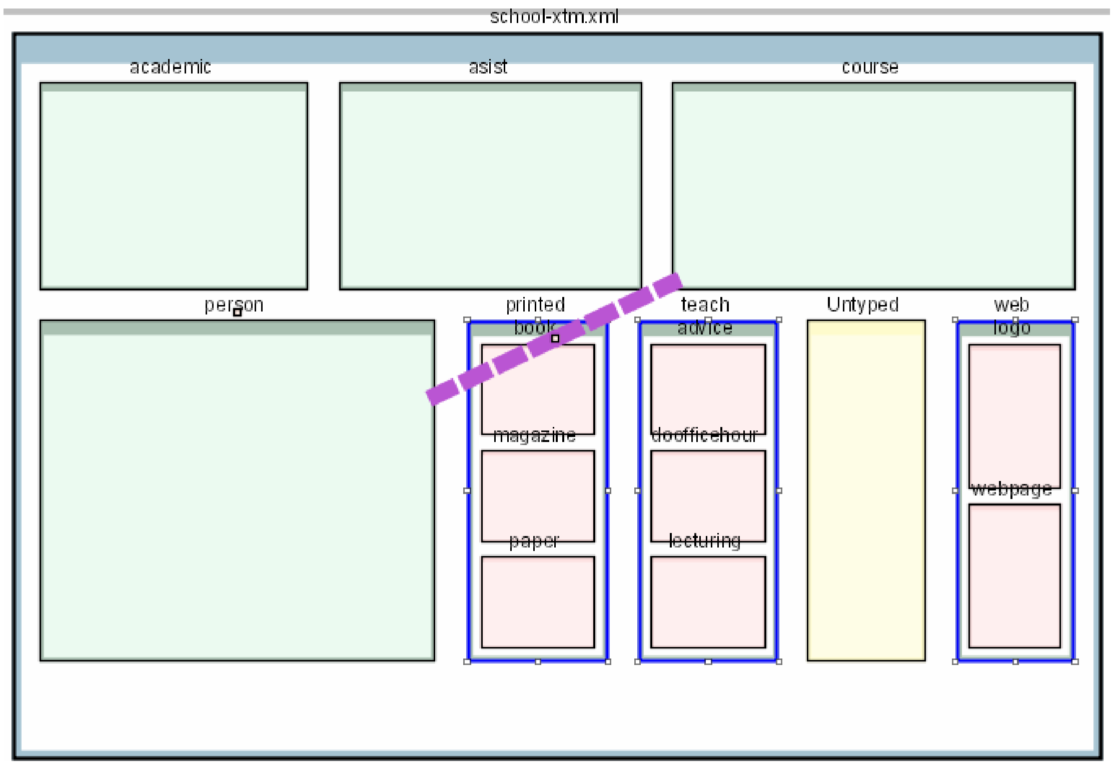


Figure 5.14: open selection operation

Close selection: Closes multiple blobs and the arcs connected to them.

pseudo code:

```
close(selectedNodes):
        for each node n in the selectedNodes
             close(n)
```

OpenAll selection: Allows opening descendants of multiple blobs and the associations connected to them. OpenAll selection operation applied to the person and course nodes is shown in Figure 5.15.

pseudo code:

```
open(selectedNodes):
        for each node n in the selectedNodes     openAll(n)
```

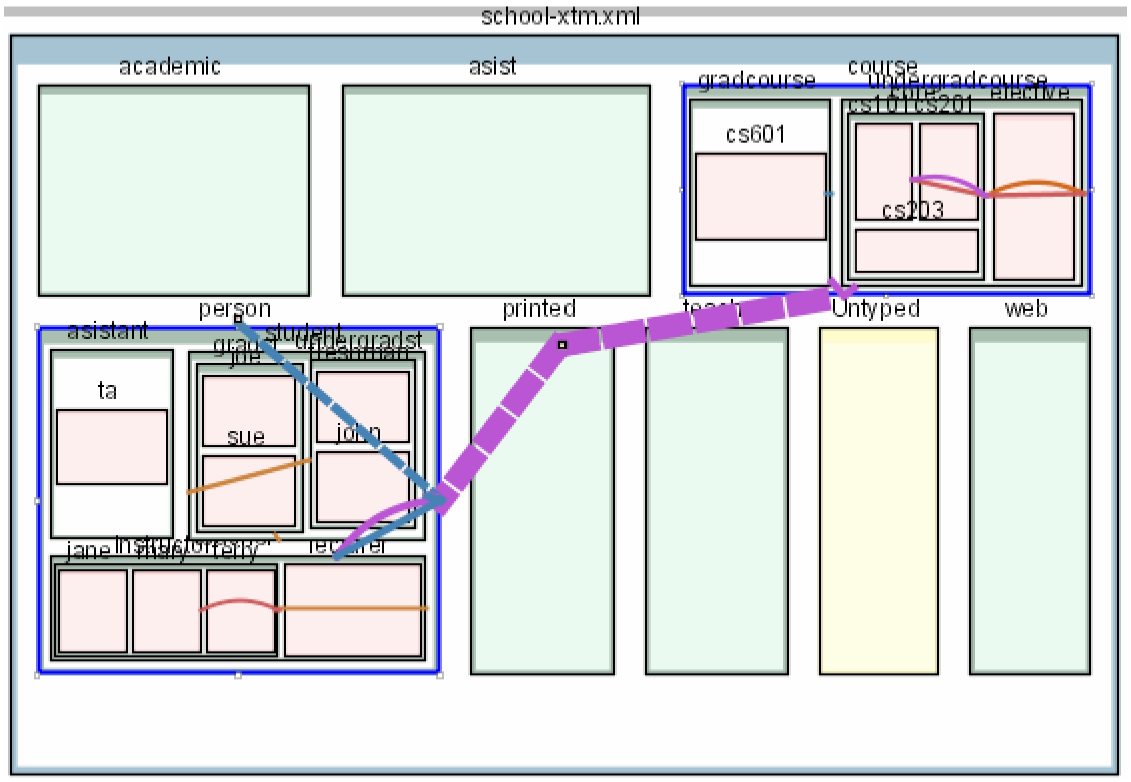


Figure 5.15: openAll selection operation

CloseAll selection: Closes descendants of multiple blobs. It also causes associations to be hidden that connected to those nodes.

```
pseudo code:
closeAll(selectedNodes):
        for each node n in the selectedNodes
             closeAll(n)
```

Group/ungroup: Allows further abstraction in current level of hierarchy by creating a new parent blob for the selected objects. The object group consists of blobs and metaedges. However they need to be connected and objects need to be siblings, in order to be grouped into a new blob. Furthermore, a set of new metaedges between the new blob and existing blobs have to be computed.

```
pseudo code:
group(selectedNodes):
      n = selectedNodes.elementAt(0)
      parent = n.getParent()
```

```
for each node n in the selectedNodes
        if n.getParent()!=parent
                dontGroup = true;
if(dontGroup)
        return
else
        Node newNode = new Node()
        newNode.setParent(parent)
        for each node n in the selectedNodes
                n.setParent(newNode)

        calculate associations in the graph
```

Following figure shows an example of group operation.

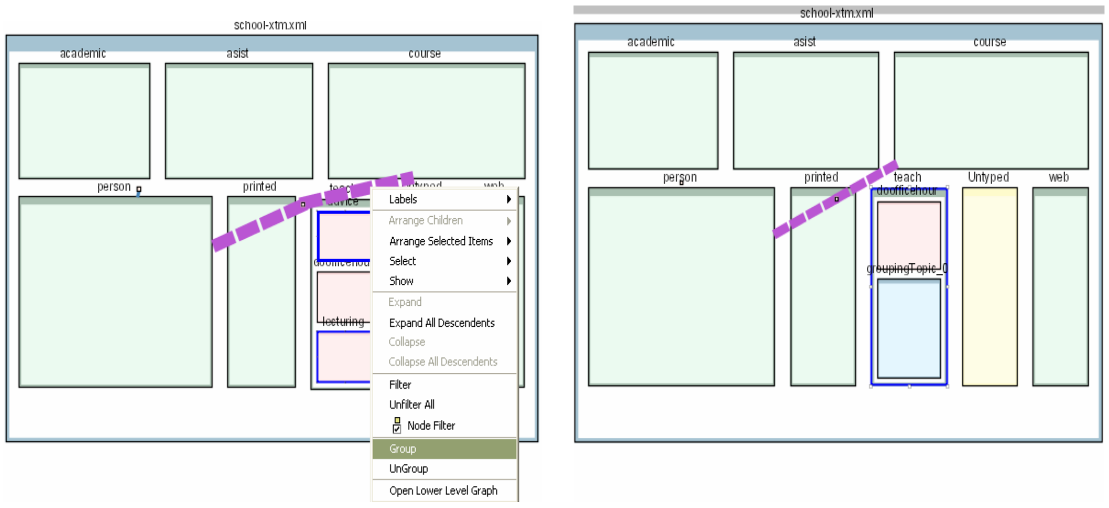


Figure 5.16: group operation

Add Untyped Node: User has an option to add a temporary type to the base topics that have no types. This simplifies the view, since those are the topics that have no children and no parent. These topics are displayed under the document root node that is at the first level of the graph. Actually, we want to give an overall view of the topic map at first levels and do not want to make the view complicated. By adding a temporary type to those topics, we push them one level lower so make them invisible until the parent node is opened. For example, they can be created as association types in the topic map and so have no children and super type. Such topics may have no importance in a TTV. If user does not want to see them at the first level of the graph, he/she uses the option to add “Untyped” type topic to the orphan topics. This temporary topic type is called “Untyped” and located as the children of root node in the hierarchy.

Zooming and Filtering: We will not go in detail of zooming and filtering operations hence they were already implemented by Shrimp.

Shrimp allows user to filter artifacts using type attributes or by selecting the ones to filter. Filtered artifacts are hidden and do not affected by further operations in the graph until user selects “unfilter all” operator. Nodes and arcs that are filtered by type attributes can also be unfiltered by choosing unfilter the selected type option.

There are “zoom in” and “zoom out” tools in the Shrimp. These operators are applied to selected node. When user select the zoom out operator and click on a node, size of the node is enlarged and it is taken into focus by covering the display. Its children are also set visible if they are invisible. “zoom in” tool make nodes smaller and allows more nodes to be visible on the screen. One zoom in operation allows view to display content of the parent of current zoomed node or the node covering the display. Further clicks on zoom in again shows the parent of current node covering the scene and its visible contents.

5.3 OTHER MODIFICATIONS TO SHRIMP

Arcs are connected to terminals at the contour of blobs in the Shrimp. Each terminal has a position and one arc is connected to one terminal. We modified PshrimpTerminal class in order to connect inner metaedgeparts to the related outer metaedges or metaedgeparts on the contour of a blob. Shrimp were creating a new terminal for a new arc. We modified the algorithm and create only one terminal on a node for related inner and outer metaedges or metaedgeparts. We also modified the size of terminals and make them smaller.

Another modification in PShrimpTerminal is to computation of terminal positions for metaarcparts. A new method computeTerminalPositionForMetaArc

(Point2D.Double srcTerminalPosition) is written to draw metaarcparts between parent and child at a position that is the nearest position between parent and child.

A new method setActiveProject(ShrimpProject activeProject) is added to the AbstractShrimpApplication class that is a base for Shrimp applications. Shrimp was able to open multiple projects at the same application but does not store the active project that is the selected therefore we could not access the active one in application. By this method we set the active project in a variable when a new project is selected by the user and can access it in any time.

addShrimpNode(ShrimpNode node) method of PNestedDisplayBean class is modified in order to add hypernodes top layer in the view. Common nodes are added under the parent nodes and can be visible only if parents are visible. However hypernodes do not have actual parents and shall be visible as soon as all member nodes are visible.

A new method hyperNodeIsInDisplay (ShrimpNode node) is added to the PNestedDisplayBean to check if a hypernode is visible or not. nodeIsInDisplay(ShrimpNode node) method of the class is also modified to call our new method for hypernodes.

AbstractShrimpViewFactor class creates shrimp view of projects. We modified it in order to add hypernodes to display at the beginning.

createShrimpArc (Relationship rel, ShrimpNode srcNode, ShrimpNode destNode) method of DataDisplayBridge.java class is modified to create different type of edges with different features. In this method, appropriate terminals are created or found for arcs; arc styles, colors and weights are set according to their types. A new method getShrimpNode (String Id) is added to the DataDisplayBridge.java class that returns the node by giving its id.

Some minor modifications are also done in the classes: RectangleNodeShape.java, GenericRigiNode.java, GenericRigiArc.java, ShrimpProject.java and ShrimpView.java.

5.4 CURRENT PROBLEMS AND FUTURE WORK

We are almost completed the implementation of all operations we listed, in the visualization tool, however we did not implement all operations we planned since we had faced with some problems. On the other hand there are additional features to implement. We can arrange all into two categories: i. Adjustments on graph and views and ii. further operations.

i. Adjustments on graph and views:

   a. Blob intersection: Supporting blob intersection to allow an object to have multiple parents. Current system supports an object to have multiple parents and display the object multiple times in the view under each parent. In ATH, RTH, and OTH multiple parents are allowed while we prevent TTH to have an object with multiple parents. For the reason that, not only the object having multiple parents is displayed multiple times in a view, the sub tree in the hierarchy rooted with this object is also copied several times. Usually a TTH has more entities than other hierarchies. So we prevent topics to have multiple parents in TTH although it is allowed in TM definition. Blob intersection solves our problem such that if an object has two parents, blobs of those parents are intersected and the child object is placed in the area of intersection as shown in the following figure.

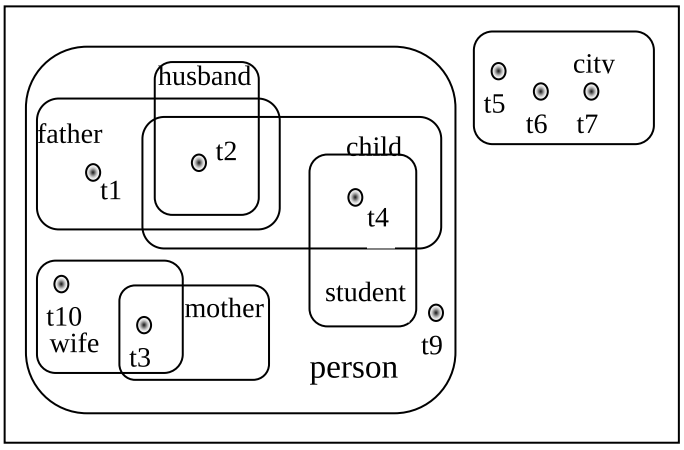


Figure 5.17: Blob intersection

b. *Overlaid Views:* Different kind of hierarchies can be viewed together. For example role type view can be drawn on the top of topic type view.

c. *Similarity of objects:* Similarity among topics, associations, and occurrences can be defined with a mathematical definition. Then distance of objects in the display may be used to represent similarity between objects.

d. *User defined weights:* We assigned weights of objects as number of descendants, that is, objects are drawn larger as the number of descendants increase. User can define own criteria for weights.

e. *Additional windows:* List of entities can be given in another window to allow user to select and highlight objects, to select a type and display the objects of this type or highlight the objects of this type in the current view, or to group/hide other details except the objects and related associations in the selected type

ii. Further operations:

a. *A Simple Query Language:* Querying of TM can be provided. User can create a query by entering values for defined attributes. For example user can say that display the objects of this type and having those number of children or having those occurrences.

b. *Advanced Filtering:* Filtering based on several threshold values can be supplied. Sample threshold values can be: # of descendants, # of metaassociations, # of objects topics in the tree rooted with the topic, # of incident blobs, average depth of the blob (This can show the complexity or generality of the topic type), average depth of the metaassociations.

CHAPTER 6

CONCLUSION

TM contain structured data describing any resources in the world. They allow users to navigate information resources at a higher level of abstractions. Although TM are well structured in order to let computers to process it, users can have problems to find relevant information within them. People often prefer less formal representation and get an overview of information space instead of diving into bare data. Since TM has multidimensional data (topics with several attributes, associations etc) and can be very large in size, therefore topic map visualization is essential. The objectives of TM visualization are helping users to navigate data, identifying relevant information and accessing it quickly.

Interactively visualizing large information spaces becomes a major research focus nowadays with recent developments in graphical processing power on desktop systems. The exploration of large data resources is an important problem as well as a difficult one. Due to the increasing number of data stored in documents, hierarchical structures get importance in organizing and visualizing documents. Scalability to visualize very large, structured documents, integrating global and local views of information space or providing different views simultaneously and to display maximum number of properties and relations in the visualization are basic requirements for a visualization tool.

In our study, we present hierarchical data model for TM. Using the class-instance relation among entities in TM we constructed 4 types of hierarchies, TTH, ATH, RTH, OTH, for 4 types of entities, topics, associations, roles and occurrences.

We developed a TM visualization tool based on multi level graphs, fundamentally based on higraphs (Harel, 1988). Multi level graphs provide different kinds of visualization of topic maps at different levels of abstractions.

Our tool allow users to navigate a topic map in a interactive hierarchical structure, while providing basic operations in information retrieval systems like zooming and filtering. We used graphical elements like size, color, shape etc to represents as much as information about data. In our tool, four different views of a topic map, TTV, ATV, RTV and OTV, are displayed at the same time.

REFERENCES

Abello, J., Dasu, T., Gansner E., et al, Graph Visualization Overview, 2001, http://www.renesys.com/projects/ leiden2001/Presentations/North.ppt

Abello, J., Korn, J., MGV: A System for Visualizing Massive Multidigraphs, IEEE Transactions on Visualization and Computer Graphics, Vol. 8, No 1, January-March 2002.

Baudon, O., Auillans, P., Graph clustering for very large topic maps. In Pamela Gennusa, editor, XML 2001, Berlin, May 2001, http://www.mondeca.com/GraphClusteringforVeryLargeTopicMaps.htm

Biezunski, M., Newcomb, S., Specializing Occurrences in Topic Maps by Association Template Subclassing, In Proceedings of Extreme Markup 2001, Montréal, August 2001, http://www.idealliance.org/papers/extreme02/html/2001/Biezunski01/EML2001Biezunski01.html

Creole (n. d.), http://www.thechiselgroup.org/creole

Davidson, R., Harel, D., Drawing Graphs Nicely Using Simulated Annealing, ACM Transactions on Graphics, Vol.15, No.4, October 1996, Pages 301-331.

Falkovych, K., Sabou, M., Stuckenschmidt, H., UML for the Semantic Web: Transformation-Based Approaches, in: B. Omelayenko, and M. Klein, ed., Knowledge Transformation for the Semantic Web, Frontiers in Artificial Intelligence and Applications, Vol.95 (IOS Press, 2003) pp. 92-106.

Fluit, C., Sabou, M., Harmelen, F., Ontology-based Information Visualization, Visualizing the Semantic Web,V. Geroimenko and C. Chen, eds., Springer-Verlag, 2003, pp. 36-48.

Freese, E., So Why Aren't Topic Maps Ruling the World?, In Proceedings of Extreme Markup Languages 2002, August 2002. http://www.idealliance.org/papers/extreme03/html/2002/Freese01/EML2002Freese01.html.

Freese, E., Taking Topic Maps to the Nth Dimension, In Proceedings of Extreme Markup Languages 2003, August 2003, http://www.mulberrytech.com/Extreme/Proceedings/html/2003/Freese01/EML2003Freese01.html

Garshol, L. M., *Topic maps, RDF, DAML, OIL: A comparison*, In Proceedings of XML 2001, IDEAlliance, http://www.ontopia.net/topicmaps/materials/tmrdfoildaml.html

Garshol, L. M., What Are Topic Maps, 2002, http://www.xml.com/pub/a/2002/09/11/topicmaps.html.

Garshol, L. M., Metadata? Thesauri? Taxonomies? Topic Maps? Making Sense of It All, 2004, http://www.ontopia.net/topicmaps/materials/tm-vs-thesauri.html

Granmo, G. O., Creating Semantically Valid Topic Maps, In Proceedings of XML Europe 2000, Paris, 12-16 June 2000, http://www.ontopia.net/topicmaps/materials/tm-schemas-paper.pdf

Gross, M. H., Sprenger, T. C., Finger, S., Visualizing Information on a Sphere, Proceedings of IEEE Information Visualization '97, Phoenix AZ, USA, October 19-24, pp.11-16, 1997.

Harel, D., On Visual Formalisms, Communications of the ACM, v.31 n.5, p.514-530, May 1988.

Harel, D., Statecharts: A Visual Formalism for Complex Systems, Science of Computer Programming, Elsevier Science Publishers B.V., 1987, pp. 231-274.

Harel, D., Yashchin, G., An Algorithm for Blob Hierarchy Layout, In Proceedings of Advanced Visual Interfaces (AVI 2000) ACM Press, pp. 29-40, May 2000.

Kal Ahmed, Topic Maps for Repositories, In Proceedings of XML Europe 2000, Paris, June 2000, http://www.gca.org/papers/xmleurope2000/papers/s29-04.html

Kal Ahmed, Developing a Topic Map Programming Model, In Proceedings of XML 2001, Florida, USA, December 9-14, 2001, http://www.idealliance.org/papers/xml2001/papers/html/05-04-03.html

Kal Ahmed, Topic Maps for Organising Information, XML UK Topic Maps Seminar, Cambridge, November 2003.

Kal Ahmed, Beyond PSIs Topic Map Design Patterns, In Proceedings of Extreme Markup Languages 2003, Montreal, August 2003, http://www.idealliance.org/papers/extreme03/html/2003/Ahmed01/EML2003Ahmed01-toc.html

Kalyanpur, A., Pastor, D. J., Battle, S., Podget, J., Automatic Mapping of OWL Ontologies into Java, The Sixteenth International Conference On Software Engineering and Knowledge Engineering, June 20-24, Canada, 2004, http://www.mindswap.org/~aditkal/www2004_OWL2Java.pdf

Keim, A. D., Information Visualization and Visual Data Mining, IEEE Transactions on Visualization and Computer Graphics, Vol.7, No.1, January-March 2002.

Kienreichh, W, Sabol, V., Granitzer, M., KAppe, F., Andrews, K., InfoSky: A system for Visual Exploration of Very Large, Hierarchically Structured Knowledge Spaces, In Proceedings of FGWM'03 - Workshop Wissens- und Erfahrungsmanagement, March 2002, http://km.aifb.uni-karlsruhe.de/ws/LLWA/fgwm/Resources/FGWM03_02_Wolfgang_Kienreich.pdf

Ksiezyk, R, Trying not to get lost with a Topic Map, XML Europe '99, April 16-30, Granada, 1999, http://www.infoloom.com/gcaconfs/WEB/granada99/ksi.htm

Ksiezyk, R, Answer Is Just a Question [of Matching Topic Maps], In Proceedings of XML Europe 2000, June 12-16, Paris, 2000, http://www.gca.org/papers/xmleurope2000/papers/s22-03.html

Kunz, C., Botsch, V., Visual Representation and Contextualization of Search Results – List and Matrix Browser, In Proceedings of Dublin Core´02, Florence, Italy, 2002, http://www.bncf.net/dc2002/program/ft/poster10.pdf

Le Grand, B., Soto, M., Information Management - Topic Maps Visualization, In Proceedings of XML Europe 2000, June 12-16, Paris, 2000, http://www.gca.org/papers/xmleurope2000/papers/s29-03.html

Le Grand, B., Soto, M., Visualization of Semantic Web: Topic Maps Visualisation, Information Visualisation, IV 2002, http://www-rp.lip6.fr/site_npa/site_rp/_publications/244-LeGrand_TopicMaps.pdf

Le Grand, B., Soto, M., Conceptual Exploration of Topic Maps, Conference Presentation in XML 2000, December 3-8, 2000. http://www-rp.lip6.fr/site_npa/site_rp/_publications/207-XML2000paper.pdf

Le Grand, B., Soto, M., Dodds, D., XML Topic Maps and Semantic Web Mining, 2001, http://www.idealliance.org/papers/xml2001papers/tm/WEB/04-04-05/04-04-05.htm

Lin,X., Soergel, D., Marchionini, G., A Self-organizing Semantic Web For Information Retrieval, in Proceedings of the 14. Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR91), Chicago, IL, October 13 - 16 1991, pp. 262-269.

Lu, Y., Hornung, J., 3D Visualization of Knowledge Domains in Learning Environments, World Conference on E-Learning in Corp., Govt., Health., & Higher Ed. 2003(1), 2285-2288, http://dl.aace.org/14137

Tollis, I. G., Graph Drawing and Information Visualization, ACM Computing Surveys, December 1996.

Marshall, M. S., Herman, I., Melançon, G., An Object Oriented Design for Graph Visualization, Reports of the CWI (Centre for Mathematics and Computer Sciences), INS-0001 (2000), ISSN 1386-3681, Amsterdam, The Netherlands, 2000. http://gvf.sourceforge.net/GVF.pdf.

Mendez, P. O., Combinatorial Hypermaps vs Topic Maps, In Proceedings of XML Europe 2001, May 21-25, Berlin, 2001, http://www.mondeca.com/CombinatorialHypermapsvsTopicMaps.htm

Miller, N., Hetzler, B., Nakamura, G., Whitney, P., The Need For Metrics in Visual Information Analysis, In Proceedings of the Workshop on New Paradigms in Information Visualization and Manipulation, November 1997, Las Vegas, pp.24-28. http://www.pnl.gov/infoviz/MetricNeedsforViz.pdf.

Moore, G., Topic Map Technology – State of the Art, In Proceedings of XML Europe 2000, June 12-16, Paris, 2000, http://www.infoloom.com/gcaconfs/WEB/paris2000/S22-04.HTM#N1

Mutton, P., Golbeck, J., Visualization of Semantic Metadata and Ontologies, In Proceedings of Information Visualization 2003, July 16-18, 2003, London, UK, http://www.cs.umd.edu/~golbeck/downloads/IV03.pdf

OLA, OWL Lite Alignment, http://www.iro.umontreal.ca/~owlola/visualization.html

Pepper, S., (n. d.), Opera Topic Map (music-xtm.xml), http://www.techquila.com/tmsamples/xtm/punk/music-xtm.xml

Pepper, S., Methods for the Automatic Construction of Topic Maps, 2002, http://www.ontopia.net/topicmaps/materials/autogen-pres.pdf

Pepper, S., The TAO of Topic Maps, In Proceedings of XML Europe 2000, June 2000, http://www.ontopia.net/topicmaps/materials/tao.html

Pepper, S., Euler, Topic Maps and Revolution, In Proceedings of XML Europe '99, Granada, April 1999, http://www.ontopia.net/topicmaps/materials/euler.pdf

Pepper, S., Navigating Haystacks and Discovering Needles, In Sperberg-McQueen, C.M.; Usdin, B.T. (eds): Markup Languages, vol.1, no.4, MIT Press, Cambridge (MA), 1999, http://www.ontopia.net/topicmaps/materials/mlangart.pdf

Piccolo (n. d.), A Structured 2D Graphics Framework, http://www.cs.umd.edu/hcil/piccolo/index.shtml

Protégé (n. d.), http://protege.stanford.edu/

Rath, H. H., Making Topic Maps More Colorful, In Proceedings of XML Europe 2000, June 2000, http://www.gca.org/papers/xmleurope2000/pdf/s29-01.pdf

Rath, H. H., Topic Maps Self-control, In Proceedings of Extreme Markup Language, 2000, http://www.gca.org/attend/2000_conferences/Extreme_2000/Papers/Rath/tom-hhr.PDF.

Rath, H. H., Pepper, S., Topic Maps: Introduction and Allegro, Markup Technologies 99, Philadelphia, USA, December 1999, http://www.ontopia.net/topicmaps/materials/allegro.pdf.

Roxborough, T., Sen, A., Graph Clustering Using Multiway Ratio Cut, In Proceedings of Graph Drawing, volume 1353 of Lect. Notes in Cornput. Sci., pages 291-296, Springer-Verlag, New York/Berlin, 1997.

Schmidt, I., Müller, C., A World to Discover: Topic Map for Thomas Mann, In Proceedings of XML Europe 2001, 21-25 May 2001, Internationals Congress Centrum (ICC), Berlin, Germany, http://www.gca.org/papers/xmleurope2001/papers/html/s10-2.html.

Shneiderman, B., The eye have it: A task by data type taxonomy for information visualizations, In Proceedings of IEEE Visual Languages , pages 336-343, Boulder, CO, Sept 1996, http://citeseer.ist.psu.edu/shneiderman96eyes.html

Shrimp (n. d.), http://www.thechiselgroup.org/shrimp

Smolnik, S., Erdmann, Visual Navigation of Distributed Knowledge Structures in Groupware-based Organizational Memories, In Proceedings of Sixth International Conference on Information Visualisation (IV'02), London, England, July, 10 –12, 2002, p. 353-360.

Smolnik, S., Nastansky, L., Knieps, T., Mental Representations and Visualization Process in Organizational Memories, In Proceedings of the 7th International Conference on Information Visualisation (IV03) - International Symposium on Knowledge Domain Visualization (IV03-KDViz), IEEE Computer Society Press, Los Alamitos CA, Washington, Brussels, Tokyo 2003, pp. 568-575.

Spoerri, A., InfoCrystal: A Visual Tool for Information Retrieval and Management, In Proceedings of the second international conference on Information and knowledge management , Washington, D.C., November 1993, pp. 11--20.

Techquila (n. d.), A Practical Introduction to Topic Maps, http://www.techquila.com/practical_intro.html

Techquila (n. d.), Topic Map Design Patterns for Information Architecture, http://www.techquila.com/tmsinia.html

TM4J, A Topic Map Engine for Java, http://www.techquila.com

Walsh, N., RDF Twig: Accessing RDF Graphs in XSLT, In Proceedings of Extreme Markup Languages 2003, August 4-8, Montreal, Quebec, Canada, http://www.mulberrytech.com/Extreme/Proceedings/ xslfo-pdf/2003/Walsh01/EML2003Walsh01.pdf

Wrightson, A., Topic Maps and Knowledge Representation, February 2001, http://www.ontopia.net/topicmaps/materials/kr-tm.html

XML Topic Maps (XTM) 1.0, 2001, http://www.topicmaps.org/xtm/1.0/ .

Vatant, B., HG4TM – Hypergraph for Topic Maps, 2002, http://www.mondeca.com/hypergraph/hg4tm.htm

APPENDIX A

XTM 1.0 DOCUMENT TYPE DECLARATION

```
<!-- ........................................................... -->
<!-- XML Topic Map DTD  ........................................ -->
<!-- topicMap: Topic Map document element ....................... -->
<!ELEMENT topicMap ( topic | association | mergeMap )*  >
<!ATTLIST topicMap
   id              ID       #IMPLIED
   xmlns           CDATA    #FIXED 'http://www.topicmaps.org/xtm/1.0/'
   xmlns:xlink     CDATA    #FIXED 'http://www.w3.org/1999/xlink'
   xml:base        CDATA    #IMPLIED  >

<!-- topic: Topic element ...................................... -->
<!ELEMENT topic ( instanceOf*, subjectIdentity?, ( baseName | occurrence )* )  >
<!ATTLIST topic     id              ID       #REQUIRED  >

<!-- instanceOf: Points To a Topic representing a class .......... -->
<!ELEMENT instanceOf ( topicRef | subjectIndicatorRef ) >
<!ATTLIST instanceOf    id              ID       #IMPLIED  >

<!-- subjectIdentity: Subject reified by Topic .................. -->
<!ELEMENT subjectIdentity ( resourceRef?, ( topicRef | subjectIndicatorRef )* ) >
<!ATTLIST subjectIdentity    id              ID       #IMPLIED  >

<!-- topicRef: Reference to a Topic element ..................... -->
<!ELEMENT topicRef  EMPTY >
<!ATTLIST topicRef
   id              ID       #IMPLIED
   xlink:type      NMTOKEN   #FIXED 'simple'
   xlink:href      CDATA     #REQUIRED  >

<!-- subjectIndicatorRef: Reference to a Subject Indicator ....... -->
<!ELEMENT subjectIndicatorRef  EMPTY >
<!ATTLIST subjectIndicatorRef
   id              ID       #IMPLIED
   xlink:type      NMTOKEN   #FIXED 'simple'
   xlink:href      CDATA     #REQUIRED  >

<!-- baseName: Base Name of a Topic ............................. -->
<!ELEMENT baseName  ( scope?, baseNameString, variant* ) >
<!ATTLIST baseName     id              ID       #IMPLIED  >
```

```
<!-- baseNameString: Base Name String container .................. -->
<!ELEMENT baseNameString  ( #PCDATA ) >
<!ATTLIST baseNameString     id              ID        #IMPLIED  >

<!-- variant: Alternate forms of Base Name ....................... -->
<!ELEMENT variant  ( parameters, variantName?, variant* ) >
<!ATTLIST variant     id              ID        #IMPLIED  >

<!-- variantName: Container for Variant Name ..................... -->
<!ELEMENT variantName  ( resourceRef | resourceData ) >
<!ATTLIST variantName     id              ID        #IMPLIED  >

<!-- parameters: Processing context for Variant .................. -->
<!ELEMENT parameters  ( topicRef | subjectIndicatorRef )+ >
<!ATTLIST parameters     id              ID        #IMPLIED  >

<!-- occurrence: Resources regarded as an Occurrence ............. -->
<!ELEMENT occurrence ( instanceOf?, scope?, ( resourceRef | resourceData ) )  >
<!ATTLIST occurrence     id              ID        #IMPLIED  >

<!-- resourceRef: Reference to a Resource ........................ -->
<!ELEMENT resourceRef  EMPTY >
<!ATTLIST resourceRef
  id              ID        #IMPLIED
  xlink:type      NMTOKEN   #FIXED 'simple'
  xlink:href      CDATA     #REQUIRED  >

<!-- resourceData: Container for Resource Data ................... -->
<!ELEMENT resourceData  ( #PCDATA ) >
<!ATTLIST resourceData     id              ID        #IMPLIED  >

<!-- association: Topic Association  ............................. -->
<!ELEMENT association ( instanceOf?, scope?, member+ ) >
<!ATTLIST association     id              ID        #IMPLIED  >

<!-- member: Member in Topic Association ......................... -->
<!ELEMENT member ( roleSpec?, ( topicRef | resourceRef | subjectIndicatorRef )* ) >
<!ATTLIST member     id              ID        #IMPLIED  >

<!-- roleSpec: Points to a Topic serving as an Association Role .. -->
<!ELEMENT roleSpec  ( topicRef | subjectIndicatorRef ) >
<!ATTLIST roleSpec     id              ID        #IMPLIED  >

<!-- scope: Reference to Topic(s) that comprise the Scope ........ -->
<!ELEMENT scope  ( topicRef  | resourceRef | subjectIndicatorRef )+ >
<!ATTLIST scope     id              ID        #IMPLIED  >

<!-- mergeMap: Merge with another Topic Map ...................... -->
<!ELEMENT mergeMap  ( topicRef | resourceRef | subjectIndicatorRef )* >
<!ATTLIST mergeMap
  id              ID        #IMPLIED
  xlink:type      NMTOKEN   #FIXED 'simple'
  xlink:href      CDATA     #REQUIRED  >

<!-- end of XML Topic Map (XTM) 1.0 DTD -->
```

APPENDIX B

SCHOOL.XTM DOCUMENT

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<topicMap xmlns:xlink="http://www.w3.org/1999/xlink">

  <topic id="person"/>
  <topic id="course"/>
  <topic id="teacher">
    <instanceOf><topicRef xlink:href="#person"/></instanceOf> </topic>
  <topic id="free1"/>
  <topic id="free2"/>
  <topic id="student">
    <instanceOf><topicRef xlink:href="#person"/></instanceOf> </topic>
  <topic id="undergradst">
    <instanceOf><topicRef xlink:href="#student"/></instanceOf>
</topic>
  <topic id="freshman">
    <instanceOf><topicRef xlink:href="#undergradst"/></instanceOf>
  </topic>
  <topic id="gradst">
    <instanceOf><topicRef xlink:href="#student"/></instanceOf>
</topic>
  <topic id="instructor">
    <instanceOf><topicRef xlink:href="#teacher"/></instanceOf>
    <instanceOf><topicRef xlink:href="#person"/></instanceOf> </topic>
  <topic id="undergradcourse">
    <instanceOf><topicRef xlink:href="#course"/></instanceOf> </topic>
  <topic id="core">
    <instanceOf><topicRef xlink:href="#undergradcourse"/></instanceOf>
  </topic>
  <topic id="elective">
    <instanceOf><topicRef xlink:href="#undergradcourse"/></instanceOf>
  </topic>
  <topic id="gradcourse">
    <instanceOf><topicRef xlink:href="#course"/></instanceOf> </topic>
  <topic id="printed"/>
  <topic id="web"/>
  <topic id="book">
    <instanceOf><topicRef xlink:href="#printed"/></instanceOf>
</topic>
  <topic id="paper">
    <instanceOf><topicRef xlink:href="#printed"/></instanceOf>
</topic>
  <topic id="magazine">
    <instanceOf><topicRef xlink:href="#printed"/></instanceOf>
</topic>
  <topic id="logo">
    <instanceOf><topicRef xlink:href="#web"/></instanceOf> </topic>
  <topic id="webpage">
    <instanceOf><topicRef xlink:href="#web"/></instanceOf> </topic>
  <topic id="john">
    <instanceOf><topicRef xlink:href="#undergradst"/></instanceOf>
  </topic>
  <topic id="joe">
    <instanceOf><topicRef xlink:href="#gradst"/></instanceOf> </topic>
  <topic id="sue">
    <instanceOf><topicRef xlink:href="#gradst"/></instanceOf> </topic>
  <topic id="mary">
    <instanceOf><topicRef xlink:href="#instructor"/></instanceOf>
  </topic>

  <topic id="terry">
    <instanceOf><topicRef xlink:href="#instructor"/></instanceOf>
    <occurrence>
      <instanceOf><topicRef xlink:href="#logo"/></instanceOf>
      <resourceRef xlink:href="http://www.anywhere.com/~terry/me.gif"
xlink:type="simple"/>
    </occurrence>
    <occurrence>
      <instanceOf><topicRef xlink:href="#magazine"/></instanceOf>
      <resourceRef xlink:href="http://www.magazines.com/~terry"
xlink:type="simple"/>
    </occurrence>
  </topic>

  <topic id="jane">
    <instanceOf><topicRef xlink:href="#instructor"/></instanceOf>
    <occurrence>
      <instanceOf><topicRef xlink:href="#paper"/></instanceOf>
      <resourceRef
xlink:href="http://www.somewhere.com/~jane/papers.html"
xlink:type="simple"/>
    </occurrence>
    <occurrence>
      <instanceOf><topicRef xlink:href="#webpage"/></instanceOf>
      <resourceRef xlink:href="http://www.somewhere.com/~jane"
xlink:type="simple"/>
    </occurrence>
  </topic>

  <topic id="cs601">
    <instanceOf><topicRef xlink:href="#gradcourse"/></instanceOf>
  </topic>
  <topic id="cs101">
    <instanceOf><topicRef xlink:href="#core"/></instanceOf> </topic>
  <topic id="cs201">
    <instanceOf><topicRef xlink:href="#core"/></instanceOf> </topic>
  <topic id="cs203">
    <instanceOf><topicRef xlink:href="#core"/></instanceOf> </topic>
  <topic id="asistant">
    <instanceOf><topicRef xlink:href="#person"/></instanceOf> </topic>
  <topic id="ta">
    <instanceOf><topicRef xlink:href="#asistant"/></instanceOf>
  </topic>
  <topic id="asist"/>
  <topic id="recitation">
    <instanceOf><topicRef xlink:href="#asist"/></instanceOf> </topic>
  <topic id="grading">
    <instanceOf><topicRef xlink:href="#asist"/></instanceOf> </topic>
  <topic id="assigngrading">
    <instanceOf><topicRef xlink:href="#grading"/></instanceOf>
</topic>
  <topic id="quizgrading">
    <instanceOf><topicRef xlink:href="#grading"/></instanceOf>
</topic>
  <topic id="midtermgrading">
    <instanceOf><topicRef xlink:href="#grading"/></instanceOf>
</topic>
  <topic id="teach"/>
  <topic id="lecturing">
    <instanceOf><topicRef xlink:href="#teach"/></instanceOf> </topic>
  <topic id="advice">
    <instanceOf><topicRef xlink:href="#teach"/></instanceOf> </topic>
  <topic id="doofficehour">
    <instanceOf><topicRef xlink:href="#teach"/></instanceOf> </topic>
  <topic id="lecturer">
    <instanceOf><topicRef xlink:href="#teacher"/></instanceOf>
</topic>
  <topic id="academic"/>
  <topic id="advisor">
    <instanceOf><topicRef xlink:href="#academic"/></instanceOf>
  </topic>
  <topic id="studentadvisor">
    <instanceOf><topicRef xlink:href="#advisor"/></instanceOf>
</topic>
  <topic id="undergradadvisor">
    <instanceOf><topicRef xlink:href="#studentadvisor"/></instanceOf>
  </topic>
  <topic id="gradadvisor">
    <instanceOf><topicRef xlink:href="#studentadvisor"/></instanceOf>
  </topic>

  <association id="sifir">
    <instanceOf><topicRef xlink:href="#advice" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#undergradadvisor" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#mary" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#undergradst" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#john" xlink:type="simple"/>
    </member>
  </association>

  <association id="bir">
    <instanceOf><topicRef xlink:href="#advice" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#gradadvisor" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#terry" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#gradst" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#joe" xlink:type="simple"/>
    </member>
  </association>

  <association id="iki">
    <instanceOf><topicRef xlink:href="#advice" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#advisor" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#jane" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs101" xlink:type="simple"/>
    </member>
  </association>

  <association id="uc">
    <instanceOf>
      <topicRef xlink:href="#assigngrading" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#ta" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#sue" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs101" xlink:type="simple"/>
    </member>
  </association>

  <association id="dort">
    <instanceOf>
      <topicRef xlink:href="#quizgrading" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#ta" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#sue" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs101" xlink:type="simple"/>
    </member>
  </association>
```

```
  <association id="bes">
    <instanceOf>
      <topicRef xlink:href="#recitation" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#ta" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#sue" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs101" xlink:type="simple"/>
    </member>
  </association>

  <association id="alti">
    <instanceOf><topicRef xlink:href="#teach" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#teacher" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#jane" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs201" xlink:type="simple"/>
    </member>
  </association>

  <association id="yedi">
    <instanceOf><topicRef xlink:href="#lecturing"
xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#lecturer" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#mary" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs601" xlink:type="simple"/>
    </member>
  </association>

  <association id="sekiz">
    <instanceOf>
      <topicRef xlink:href="#midtermgrading" xlink:type="simple"/>
    </instanceOf>
    <member>
      <roleSpec>
        <topicRef xlink:href="#instructor" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#jane" xlink:type="simple"/>
    </member>
    <member>
      <roleSpec>
        <topicRef xlink:href="#course" xlink:type="simple"/>
      </roleSpec>
      <topicRef xlink:href="#cs601" xlink:type="simple"/>
    </member>
  </association>

</topicMap>
```