THEORY AND PRACTICE OF THE TERNARY RELATIONS MODEL OF INFORMATION MANAGEMENT

By

Amir Pourabdollah

MSc., BEng.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

February 2009

ABSTRACT

This thesis proposes a new, highly generalised and fundamental, information-modelling framework called the TRM (Ternary Relations Model). The TRM was designed to be a model for converging a number of differing paradigms of information management, some of which are quite isolated. These include areas such as: hypertext navigation; relational databases; semi-structured databases; the Semantic Web; ZigZag and workflow modelling.

While many related works model linking by the connection of two ends, the TRM adds a third element to this, thereby enriching the links with associative meanings. The TRM is a formal description of a technique that establishes bi-directional and dynamic node-link structures in which each link is an ordered triple of three other nodes. The key features that makes the TRM distinct from other triple-based models (such as RDF) is the integration of bi-directionality, functional links and simplicity in the definition and elements hierarchy.

There are two useful applications of the TRM. Firstly it may be used as a tool for the analysis of information models, to elucidate connections and parallels. Secondly, it may be used as a "construction kit" to build new paradigms and/or applications in information management. The TRM may be used to provide a substrate for building diverse systems, such as adaptive hypertext, schemaless database, query languages, hyperlink models and workflow management systems. It is, however, highly generalised and is by no means limited to these purposes.

PUBLISHED PAPERS

The research of this thesis has led publishing the following papers, copies of which can be found in Appendix D.

1. Pourabdollah, H. Ashman, and T. Brailsford, "Are We Talking About the Same Structure? A Unified Approach to Hypertext Links, XML, RDF and ZigZag," in *Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia*. Pittsburgh, PA, USA: ACM, 2008.

2. Pourabdollah, T. Brailsford, and H. Ashman, "A User-Oriented Design for Business Workflow Systems," in *Lecture Notes in Computer Science*, vol. 4473, D. Draheim and G. Weber, Eds. Berlin Heidleberg: Springer-Verlag, 2007, pp. 285-297.

3. Pourabdollah and M. Hartley, "Gathering Unstructured Workflow Data into Relational Database Model Using Process Definition Language," in *Proceedings of IASTED International Conference on Databases and Applications*. Innsbruck, Austria: ACTA Press, 2006, pp. 32-37.

4. Pourabdollah, T. Brailsford, and H. Ashman, "A User-Oriented Design for Business Workflow Systems," in *Proceedings of the 2nd Conference on Trends in Enterprise Application Architecture*, 2006.

5. A. Pourabdollah, T. Brailsford, and H. Ashman, "Relations Modelling Sets of Hypermedia Links, Navigation and Semantics", *the Computer Journal*, (submitted after revision, January 2009)

ACKNOWLEDGMENTS

I wish to use this opportunity to thank those who supported me during this research.

First of all, many thanks to my dear supervisors Dr. Tim Brailsford and Dr. Helen Ashman for all of their outstanding and invaluable support to this research and to my educations.

I would like to thank all of my colleagues in the Web Technologies Research Group for their support and advices to this research, in particular Dr. Liz Brown, Dr. James Goulding and Dr. Adam Moore. Also thanks to Dr. Peter Blanchfield for his comments on my first year report which helped me in developing the rest of this research.

Many thanks to the administration of the University of Nottingham, in particular to the staff of the International Office who kindly offered me the Research Tuition Fee Scholarship, to the staff of the Graduate School in particular Dr. Sue Scarborough, and to the staff of the School of Computer Science in particular Mrs. Karen Attwood. This research would not be possible without their support.

I would also thank the examiners for this thesis, Dr. Alexandra Cristea and Dr. Peter Blanchfield for their reading and comments.

Finally special thanks to my family, particularly to my dear wife Motahare for all of their extraordinary supports and patients.

TABLE OF CHAPTERS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF ABBREVIATIONS

The following abbreviations have been used in this thesis for the first time:

- TRM: Ternary Relations Model: A formal description of building relations by three nodes of data.
- TRM-DB: TRM Database, a database design method utilizing the TRM.
- TRM-NAV: TRM Navigation, a hypertext navigation framework having TRM-based hyperlinks.
- TRM-WF: TRM Workflow, a model of describing workflows by the TRM.
- TWM: TRM Workflow Management, a workflow management system based on the TRM-WF.
- pE: predicate-Expression, a form of making binary links in which the link source is selected by a logical predicate and the destination by a computation.
- pfE: predicate-function, Expression, the extension of pE to be used in ternary links, in which the association of a link is also computed by a function.

Among many other abbreviations used from other related works, these are the most important and/or less known acronyms which are described throughout this thesis:

RDF: Resource Description Framework, OWL: Ontology Web Language, UML: Unified Modelling Language, BPMN: Business Process Modelling Notation, XPDL: XML Process Definition Language, YAWL: Yet Another Workflow Language, FOHM: Fundamental Open Hypermedia, COHSE: Conceptual Open Hypertext Service, IUHM: Information Unit Hypertext Model, MMVP: Model Map View Praxis, WFMC: Workflow Management Coalition, W3C: WWW Consortium.

To My Dear Parents.

Chapter 1 -

INTRODUCTION

"Generalization" or "Unification" is what many scientists from many fields of human knowledge have targeted, particularly in the theoretical sciences, from the recent attempts of physicists like Stephen Hawkins [96] to the universal model of Ken Wilber [200] to explain almost "everything" in this world. The philosophical motivation for those scientists to look for such a global theory is that in a universal scale, there must be only one truth that can explain all the true facts in this world.

The domain of this thesis is much smaller than such universal theories, moreover, it is not as purely theoretical as those. Also the unification in this thesis is neither an achieved nor a targeted goal, but it is a just an approach to target or achieve some results. However, the motivation is still the same.

If a unified theory of physics can explain an existing physical phenomenon and predict the happening ones, then by analogy, a unified information model in IT domain may help to explain the similarities between existing systems and to build new systems.

1.1 Background

Early computers were only for computing but now they are an integrated –if not essential- part of human life. What the today computers can do over the traditional computing is managing "information", and most of the today computer systems are in fact, "information management" systems. Before providing the dealt problems, questions, aims and objectives of this thesis, it is necessary here to have a touch of the challenging world of "information" terminology in the computer science's literature. Also the importance of a unifying approach will be explained to address some problems and answer some fundamental question in the information management community.

1.1.1 Information and information Model

In the field of information science and knowledge management, the terms data, information, knowledge and wisdom are often considered to make a DIKW pyramid or hierarchy proposed by Russell Ackoff in 1989 [11, 157, 166], as illustrated inFigure 1-1.

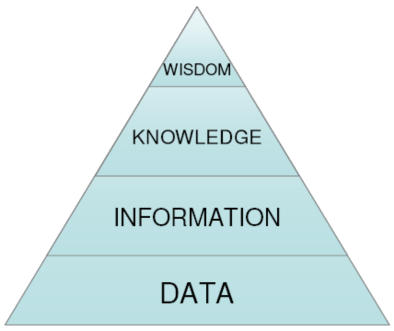


Figure 1-1: DIKW pyramid [97]

For the description of data as the basic layer, Ackoff's view in [11] is that *Data is raw; it simply exists and has no significance beyond its existence (in and of itself). It can exist in any form, usable or not. It does not have meaning of itself.*

On top of the data layer, "information" adds context to data, knowledge adds "how to use" to the information. The term of information may be used in different contexts in the field of computer science, and thus it is very difficult to find such a general definition. The main thing that makes information different from any kind of raw data is the role of human perception. Information is sometimes defined as *anything that a human being can be interested in* [125], or as *any represented pattern* [98]. The term of knowledge is particularly difficult to define, as reviewed in [174]. In that review, a working definition of knowledge is proposed as *the high-value form of information that is ready to apply to decisions and actions*. For the purposes of this thesis, knowledge is considered to be the *meaningful structure of information.*

Briefly, Information is the structured form of raw data that can be interpreted or put into some context. Knowledge is the interpreted and meaningful structure of information that can be use to make decisions. Wisdom then adds "when to use" elements and the decision making and reasoning skills in a time-dependent context on top of the gathered knowledge.

Also in [85], the layer of "understanding" is inserted between wisdom and knowledge layers, which includes the analyze and synthesize processes in order to reason or make a decision or reasoning.

It may be useful to illustrate the above hierarchy in Figure 1-2.

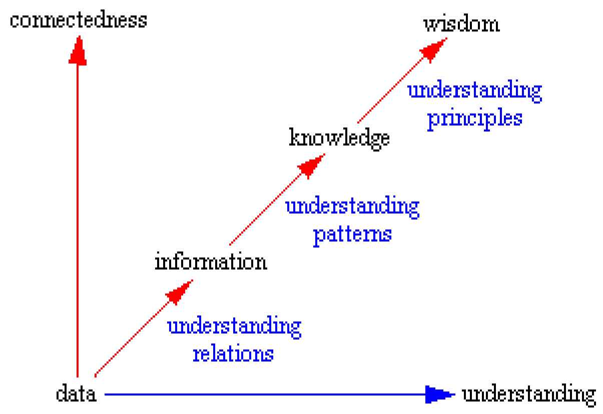


Figure 1-2: Transition between data, information and knowledge in DIKW hierarchy [85]

Some related works have criticized DIKW hierarchy to be not a perfect explanation. An opposite idea is that the relation between data, information and knowledge is not directional and hierarchical at all. As evidence, building relations to make information out of data is not possible without knowledge, and that the knowledge is not meaningful without knowing the "knower" [43]. Another work [75] focuses on the "holistic" nature of data, information and knowledge and that the relationship between them may be explained by "meta"s not by linear dependencies.

According to the above reviews, defining data, information and knowledge is a challenging argument. It may be impossible to draw strict lines between data, information and knowledge and they are interchangeable in many contexts. As evidence, a structure of information may be itself an information building block in another context.

In this thesis, the DIKW hierarchy with some sort of flexible and fuzzy borders is generally accepted. It is also useful to have a working definition for the term of "Information model". It is defined here as *the conceptual and/or logical way that a computerized system uses to transit from data to information, as defined in DIKW hierarchy*. According to the DIKW illustration of Figure 1-2, this definition is almost equal to "how a system understands the *relations*".

1.1.2 Information Modelling and Relations

The defined "information model" exists in any information management system (practically) or information management paradigm (theoretically). This thesis considers a number of information management paradigms to be candidates of the mentioned "information model unification". Since the concept of "relations" has shown to be the main issue in an information model, then the unification is mainly about unifying how to relate pieces of data to make information (or in a wider definition, to relate simpler pieces of information to make more complex ones).

It is noticeable that the term "relation" has also been used in the Relational Databases Theory in a set-theory mathematical context [58] meaning a collection of related records of data, like a table [1]. This shall not be confused with the "relation" term used in this thesis hereafter. This term is used in this thesis for its pure meaning, as something having relationship, as in relating things together.

It is possible to rethink some of the known information modelling paradigms, in terms of what and how they relate together, as follows:

1. Making relations between pieces of data to make tables and relating tables to build databases.
2. Making relations between nodes of information by hyperlinks, to build a hypertext system.
3. Making relations between pieces of data and/or metadata in a textual and semi-structured manner to build XML listings.
4. Making semantic relations between Web resources to build the Semantic Web.
5. Making relations between cells in a multi-dimensional hyperspace to make ZigZag.
6. Making relations between tasks, actions and decisions to build a workflow management system.

[1] In that reference, Codd defines the term "relation" as *Given sets S1, S2, … , Sn (not necessarily distinct),* R *is a relation on these n sets if it is a set of n-tuples each of which has its first element from S1, its second element from S2, and so on. More concisely,* R *is a subset of the Cartesian product S1 X S2 X … X Sn*

More details on each of the above systems will be provided in chapter 1, but they have been counted above to show the motivation of selecting these systems to be studied in this thesis. These systems may be mostly different in terms of look and applications, but they are similar in establishing some relations.

1.1.3 Questions and Problems

When researchers from different domains of the information management community get together in conference bars, one of the main topics of debate that almost always comes up shortly after the "my system is better than your system" conversation, is the "your system is really the same as my system" conversation. For example, people who work on ZigZag are often told, in no uncertain terms, that ZigZag is "merely" a different take on the Semantic Web or that it is XML in heavy disguise.

Although many of the major paradigms of the information management superficially look to be very different, on a deeper level they do have a lot in common – they are addressing many of the same issues, and utilising many of the same techniques to do so. They all divide information into independent pieces of data (a set of nodes) and they all associate these nodes with each other (a set of links). This is called "node-link structure" in this thesis.

The simplest node-link structure is the binary model, where each two nodes can be simply connected by means of a link. However, a limitation of the binary model is its inability to express the purpose of a link, either for human or for machines. This is not essential for the technical implementation, as evidenced by the number of systems that offer links without any indication of their purpose (the Web being a prime example), but the purpose of the link, its reason for being and its semantic implications are nowhere represented in the binary model. Knowing that two piece of information are connected -without knowing by which mean they are connected- may not be enough to transit between data to information, and from information to knowledge in DIKW hierarchy (section 1.1.1). This will be explained more in chapter 2. As an example, without any indication of why a link should be followed in a hypertext system, a user could easily waste time exploring irrelevant links. In *extremis* he or she might even give up on the hypertext and go to a search engine for a more rapid answer to his or her needs.

There are many information management paradigms such as ZigZag and the Semantic Web which incorporate an awareness of "the why" of a link. These sorts of systems are more likely to fit into DIKW hierarchy. It will be shown in chapter 2 that more enriched links (with typing, semantic elements, etc.) can produce more knowledge-oriented systems.

A logical successor to the binary model is to use a ternary approach – where the relations consist of not just the two linked nodes but also a third node, which represents the link between them. The ternary version of the node-link structure, which is developed in this thesis, is exemplified by a link that connects three arbitrary nodes in an ordered manner. So, a link originates from a node, passes through another node and is terminated in a third node. This ternary approach is not new, but it has not previously been used in this way to unify the fundamentals of different paradigms in the information management community. A known special case to such a node-link structure is the directed labelled graph, whereby a separated node of information is demoted to a label and is used just for describing a binary link. An obvious difference is that a label can no longer be involved in any other link.

As a summary, the focused problems are:

1. Some information paradigms have limitations to completely fit into DIKW hierarchy.

2. Many cases of isolating an information system paradigm from others exist because of ignoring their commonalities.

And the main questions are:

1. Is there any generalized information model that firstly can satisfy DIKW and secondly the studied different modelling paradigms are considered to be special cases of that?

2. Finding that model, can some new solutions be found to communicate between the studied information systems in their information modelling level?

3. Knowing that different paradigms are special cases of the found model; can some new information management paradigm be thought to be directly based on it, particularly using the most of the found model?

By an analogy to the unified force theory [96] that physicians are developing, the above three questions may have some equivalents like: 1) Can a unified force be found that gravity, electromagnetic and atomic particle forces are special cases of that? 2) Can those three fundamental forces be interchangeable; and 3) Can some new physical system be thought that uses that "X-force" directly?

1.2 Aims

According to the explained approach in the previous section, the aims of this thesis are:

1. *To find a unified and simple information model for the different studied information management systems.*

2. *To investigate how the found information model can be used to bridge over some known information management systems.*

3. *To investigate the potential of the found model in making new paradigms and/or information management systems, that may not be known or formalized before.*

1.3 Objectives

For achieving the aims mentioned in the previous section, the objectives are considered to be:

1. *A top-down study method*: To have a unifying approach in studying the related works in a knowledge management context. The studied related works are:

    a. The Relational Databases

    b. The Semi-structured Databases and XML

    c. Hyperstructure Links

    d. The Semantic Web

    e. ZigZag

    f. Workflow Definition Models

    The main reason for selecting the above set of related models is their similarity in fitting to a node-link structure (as explained in section 1.1.2). They are all the commonly used paradigms in the hypertext and information management communities. The model developed in this thesis does not rely on the specifications of the above six approaches – however it will be shown that it defines an abstract model which underpins the above paradigms, and indeed any other node-link structure approach in information management.

During this related work study, different thoughts, advantages and disadvantages will be studied, together with some unifying "ternary-based" rethinking. Also the concept of "knowledge-orientation" will be provided to investigate each paradigm in its potential to be used as a "knowledge transfer media". Particularly, in studying the hypertext links, different approaches to use hyperlinks to build knowledge-oriented hypertext will be reviewed.

2. *Forming the TRM:* To define and formulate the found fundamental model as the "Ternary Relations Model" or "TRM" in the most generalized way so it can cover the studied information models. It will also be considered that the targeted information model is not known from the beginning and also it may or may not be generalized in a wider context than the studied systems. TRM will be defined as static and dynamic versions and for each one a formal description will be provided. The definition of TRM shall be both very simple and very general to be able to be a useful unification. Moreover, a layered approach will be proposed to explain how TRM fits in with other related works.

3. *A bottom-up study method:* To introduce three new information management threads on top of TRM, as follows:

    - *A New Schemaless Database Paradigm*

    - *A New Hypertext Navigation Model*

    - *A New Workflow Definition Model*

    The TRM in this part is considered to be an "information model construction kit" and this group of objectives is directed to search for the ability of the TRM to build new paradigms and systems. For each one of the above, it will be shown how TRM can partly or wholly be used to build new systems.

For the first thread, the problems of associating strict schema in structured and semi-structured databases and the imposed limitations to handle real-life information will be studied, then by using the TRM formulation, two methods of using tables or XML to build databases without any associated schema will be demonstrated. A TRM specific query language will be also proposed together with its implementations in SQL and XQuery.

For the second thread, TRM is considered to be an extension to binary navigational models, in which the concept of "binary links" may be extended to "ternary links". Through demonstrating a developed system, it will be shown how ternary links can be used to enrich or adapt hypertext systems. Also the similarities and differences between the Ternary Relations Model and RDF data model used in the Semantic Web will be discussed.

For the third thread, workflow definition model is considered to be a new area to apply TRM theory to, especially by considering dynamic and bi-directional properties of TRM. Also a demonstration of a workflow system developed on top of TRM will be provided.

1.4 Structure of the Thesis

According to the three objectives mentioned in the previous section, chapter 2 is the place of the top-down study method, chapter 3 is the place of forming TRM and chapters 4 to 7 are the places for the bottom-up study method.

During the bottom-up study, chapter 4 develops the New Schemaless Database Model, chapter 5 develops a New Hypertext Navigation Model and chapter 6 develops a New Workflow Definition Model. Finally chapter 7 uses the idea of chapter 6 in a system development case study.

The discussion in chapter 8 reviews the TRM development and practice in an integrated method to reach the final conclusion about the rationale of the TRM and to overview the possible future works.

1.5 Contributions

The contribution of this research can be listed as follows:

1. Extraction and formalization of the TRM as a substantial information model, providing a tool to define, interconnect and analyze different information management approaches for the first time.

2. The applications of TRM in hypertext navigation, by proposing a new modelling framework for hyperlinks.

3. The applications of TRM in database theory, by introducing a new class of schemaless databases.

4. The applications of TRM in the workflow management systems, by defining a new extended framework for defining workflows, supported by practical systems.

This thesis provides its main "product" as a "model" together with a set of design ideas for information system designers and users, as categorized in the above 4 items. These design ideas which are mostly theoretical can help making new system benefiting from the advantages of the developed information model, i.e. interfacing between existing model and using its extra conceptual features. It is noticeable than the practical works in this thesis are mostly for demonstrations purposes and must not be considered as the final products of this thesis.

C h a p t e r   2 -

THE RELATED WORKS: LOOKING FOR A COMMON FOUNDATION

In this chapter five major paradigms of current information management thinking – the Relational Databases, Semi-structured databases and XML, the Semantic Web, ZigZag, and Workflow Models– will be reviewed and all will be directed to a "ternary approach". It will be shown that a special kind of node-link structure can be similarly found in those various information management paradigms. By doing so an attempt will be made to answer the question of "whether we are indeed all talking about the same fundamental structure".

2.1 The Relational Databases

Today the relational databases are the most common way of using computers to store and retrieve information. The relational databases are based on a mathematical model introduced by Codd in 1970 [58]. From that time till now, the relational database could practically integrate or overlay the existing approaches about how information can be stored in "tables". Tables consist of rows and columns, which is an intrinsically "rigid", and hence inflexible, Cartesian structure. Nevertheless, the paradigm of the relational databases provides one of the most known and consistent method of data management. The main components in the theory of the relational databases [55, 56, 58] are:

1- Relations (a set of interlinked tables) and a set of Constrains applied on them.

2- Normalization (A formal method about how to optimally design tables and their links in order to meet integrity constrains and to avoid redundancy). Details can be found in many database textbooks like in [113])

3- A sub-language or algebra (to provide the necessary language for a formal approach to store, modify and retrieve information).

Formally, a relational database has a schema, in which it is specified how the tables are designed and interconnected. The database schemas are expressed using a format like $r_1(a,b)$, $r_2(c,d,e)$ where r is the name of the table (also called *relations* in the context of the relational databases) and other letters show the column names.

Linking the tables in an optimal way is what makes the relational databases distinctive from any other database system. This is called "relational processing" in the Codd's theory and a database system that does not support it should be considered as non-relational. Interestingly, Codd uses the term of "navigation" to express the systematic functionality for linking between tables: "There is a large difference in implementation complexity between tabular systems, in which the programmer does his own navigation, and relational systems, in which the system does the navigation for him, i.e., the system provides automatic navigation" [57].

2.1.1 The Common Challenges

The first challenging issue about the relational databases comes back to the fundamental characteristic of tables, which are a rigid structure or rows and columns. They are good when one precisely knows which data fields are required for expressing a piece of information, but not the best choice for dealing with the irregularity and the dynamic properties of the real-life information.

A certain set of data fields in tables may not be adequate for expressing much real-life information. To overcome this problem, two approaches may be used. The first approach is to design as many tables as necessary, each with a different data field design to fit a group of information, and finally to link these tables. As the number of necessary information structures increases in the real-life cases, the tables converge to simpler structures having less data fields, and the number of tables increases. The *extreme* point of this approach is the binary decomposition, where the databases is being normalised down to numerous binary tables. This process, which is called "binary decomposition" –also known as 6th normal form- is theoretically possible but practically difficult to manage, because the resources of a database management system will be highly allocated to manage numerous links between numerous tables rather than being allocated to data storage and retrieval tasks. As the second approach, one may want to keep the number of tables limited, so he/she may design them in a maximal method (predicting all possible data fields in single tables). The *extreme* point of this approach is ending up with a few but large non-normalized and non-relational tables which consequently increasing the redundancies and null values.

For several decades, any small change in schema design could have been a serious problem for database developers, especially for working database systems. No one can stop the dynamic and irregular aspects of the client requirements, so the mentioned problem can always happen. This may imply deep changes in a working system with all the risks of data or efficiency loss. This seems to be a built-in problem of relational databases, something that completely comes back to the rigid nature of database tables.

Null values are also another challenging matter in RDBs. Nulls may cause ambiguity because they neither express anything when a piece of data is expected, nor specify which of the possible cases have happened: "unknown" or "nothing"? By definition, a record (or a tuple in RDB terminology) is a complete piece of information in the context of the container table structure. Either the null value is interpreted as "nothing" or as "unknown", a tuple containing a null value handles an imperfect piece of knowledge and thus cannot be a tuple by definition.

Although normalization can provide a method of avoiding null values when it is expected to happen, but no database designer can guarantee the availability of all required fields in the decomposed tables at data entry time because "unknown" or "nothing" can always happen in the real life. Also normalising down to a set of binary pairs may produce a numerous number of joined relations which may be impractical to manage. That is why fields of RDB tables are -by default- ready to accept null values except for the ones tagged as "required". If one needs to completely avoid null values, he or she will again end up with a binary decomposition version of the database.

Also "null values" are not "values", but something about values (which is the lack of a value). By this view, a null is naturally meta-data. The question then will be "can a meta-data sit in a tuple?" A tuple is defined in RDB theory as a set of tagged data and not tagged meta-data. Thus a meta-data in a tuple cannot theoretically fulfil the information gap, and the answer to that question is negative.

Different approaches to the null values in RDB community has been taken in order to fit the null values in the RDB theory, either by changing the interpretation of the null values, or by slight changes of the RDB theory. Codd himself extended his RDB theory to include "unknown" values, i.e. those which are existing in the real world but we don't know them [55]. A main problem of this approach is violating the Set Theory, which is the basis for RDB theory: Simply, if a null (as defined in unknown interpretation) is a member of a set, then is that set equal to itself? The answer is not a definite yes because that null value is not a static member. A newer version of this approach is building RDB theory based on a fuzzy-set theory in order to estimate null values, like in [50]. Another approach is interpreting nulls as "non-existence" and the problem is again changes to dealing with "information incompleteness" like in [90, 108]. Null values are even interpreted as the combined values of "unknown or non-existence" like in [178] or "no-information" in [202]. These approaches are still suffering from the mentioned problem of mixing data and meta-data. An in-depth meta-data approach to the missing/incomplete information (of any interpretation) has been studied in [143]. Also Date and Darwen in [62] provide ideas against null values and Darwen in [61] proposes how to practically avoid them in databases.

From the point of view of this research, it doesn't so much matter to know whether or not the null values can finally fit in the RDB theory. Instead, it is important to know about the presence of such research challenges and why it is preferred to avoid the null values in the information model provided in this thesis.

2.1.2 An Example

The example shown here is a simple bibliography database including some journal articles. This example will be re-used in the rest of this thesis on various occasions. Here after explaining the example, the non-normalized and normalized versions of implementing it in the relational databases will be shown.

The database includes some journal articles; each article has some authors, a title, a journal name and a year of publication. For each article, the number of authors can be 0 to many, and the other fields are necessarily single. An example with three articles is shown in Table 2-1 (the non-normalized form).

Table 2-1: The sample database, non-normalized form

Title	Author	Journal	Year
Enterprise-Wide Workflow Management	C. Bussler	IEEE Concurrency	1999
On the Evaluation of Workflow Systems in Business Processes	S. Choenni R. Bakker	Journal of Information Systems Evaluation	2003
Searching for e-Business Performance Measurement Systems	Null	Journal of Information Systems Evaluation	2006

The two problems of the non-normalized form can be observed as redundancies (like the repeated name of journal) and null values (like the author name). After the normalization process, the tables below are produced. Using the unique identifier fields, the normalized form avoids redundancy and null problems.

Table 2-2: The sample database, normalized form in 4 tables

ArticleID	Title	JournalID	Year
ArticleID1	Enterprise-Wide Workflow Management	JournalID1	1999
ArticleID2	On the Evaluation of Workflow Systems in Business Processes	JournalID2	2003
ArticleID3	Searching for e-Business Performance Measurement Systems	JournalID2	2006

AuthorID	Author
AuthorID1	C. Bussler
AuthorID2a	S. Choenni
AuthorID2b	R. Bakker

ArticleID	AuthorID
ArticleID1	AuthorID1
ArticleID2	AuthorID2a
ArticleID2	AuthorID2b

JournalID	Journal
JournalID1	IEEE Concurrency
JournalID2	Journal of Information Systems Evaluation

2.1.3 The Relational Databases: A Ternary Approach

There are two possible ways of ternary approaches to the relational databases.

First, a table can be considered as a set of finite triples: (row number, cell content, column name). The constrains representing the links also are triples of (table name, joining field, table name). Although this approach has so much redundancies by repeating row number and column name for each piece of data, but it is theoretically enough to show that a table is built on a ternary node-link structure .

The second approach is motivated by binary decomposition rules [55]. As shown in the example, it is proved that a relational database can be decomposed to a set of finite linked tables, each with two columns. Having this, the entire database is convertible to triples of (first cell content, table name, second cell content). Again, although this is not practically useful to fully decompose a database, the theory is enough for the target of this research. This will be re-explained formally after defining the TRM in section 4.1.1.

2.2 Semi-Structured Databases and XML

The general term of "semi-structured databases" refers to a group of approaches that try to avoid the fundamental regularity of tables (described in section 2.1.1). They also have been called "Schemaless" or "Self-describing" databases [10]. However, the term is very difficult to define, because what it is not is clearer than what it is.

When merging databases from different origins started to become unavoidable in the recent ten years, especially when the Web facilitated that integration, the term "semi-structured" was referring to some solutions to avoid rigidity of tables in information management. When XML was introduced in 1998 by W3C, it soon became the most common way to express information in a "semi-structured" manner [10, 171]. The obvious advantage that makes it common is its wide acceptance as a standard of data exchange on the Web, thanks to the textual basis of the language and the easiness of text-processing. The XML's simplicity, together with its readability by both humans and machines helped to make it a global standard, and also to be surrounded by a confusing number of XML-based standards and languages, such as RDF-XML, XML-Schema, XHTML, etc. XML also showed its other major characteristic: There is no separated description of the structure; i.e. XML describes its content internally, thus the term "Self-describing" has also been used for XML [101].

In RDB tables, the meaning of data (or meta-data) is expressed in the schema (tables design), so a piece of data is interpreted by knowing its location in a certain row, a certain columns of a certain table. The good side of this is that if the tables are designed optimally, the space required for the database is optimally low, because the schema is stored once and serialized data can be mapped into the schema easily.

In using the semi-structured approaches, the data is described by mixed and repeated meta-data which has a cost of increasing storage space. The "separated" meta-data is now changed to some "joint" ones. For example, the labelled graphs are some means of the semi-structured data, in which labels carry meta-data and nodes carry data.

The waste of the storage space is a dark-side of XML which is usually ignored thanks to the memory technologies. Most of the XML features are common with semi-structured data concepts. However, XML has its own set of problems which the research on semi-structured data has not yet addressed, considered important or solved [171]. The details of these differences are beyond the scope of this review. In this thesis XML is generally used as a language to express the semi-structured data.

In XML, a general syntax is like:

```
<element attribute="xxx">
    <sub-element…>
        yyy
    </sub-element>
</element>
```

"Elements" carry the meta-data part of the database, either by name of the elements (also known as "tags"), or by name of the attributes. Data are inside the elements, either as the attribute values (like xxx above), or the element values (like yyy above). Elements can have sub-elements with all the properties of an element, so XML is equivalent to a tree of hierarchical elements. The same structure can be shown as a directed labelled graph [175] like the illustration of the above example in Figure 2-1. Some slightly different strategies for this conversion have been described in [175].

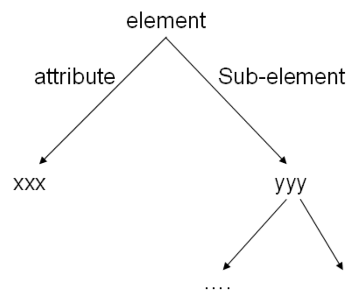


Figure 2-1: Sample directed labelled graph for XML representation

A XML listing is completely self-describing. For consistency purposes it is usually preferred to "validate" an XML listing by certain rules. The names of the elements and attributes and their hierarchical interconnection are stored separately (using DTD or XMLSchema methods) so a mechanism can be used to check the consistency and to qualify a XML listing against the specified rules. More details about DTDs, XMLSchemas and XML query languages are out of the scope of this thesis and can be found on the World Wide Web Consortium website (www.w3.org).

2.2.1 The Common Challenges

Using the term "semi-structured" for XML leads to assume the existence of both "structured" and "non-structured" aspects for XML. The non-structured aspect (or being "schemaless") is because of XML's self-descriptive characteristic and that it doesn't necessarily need an external meta-data to become expressive. According to the "structured" aspect, XML has characteristics like hierarchy; i.e. building a tree of information and putting each piece of information on some nested levels of that tree. According to Ted Nelson's view [132], the existence of hierarchy is a classical property of many computer systems, and is originated by the paper-based look to the computers which may prevent a computer system from being more extensible and scalable to be used in the real-life applications. Also, XML might be validated and for a validated XML, an external schema is required and the self-descriptive characteristic is no longer exists. These two issues can potentially threat on the flexibility of the resulted database systems. However, it is still absolutely possible to build XML with single level of hierarchy and without validation requirement. This type of XML is what will be used in chapter 4 as a storage layer for TRM, called TRM-XML.

2.2.2 An Example

Recalling the bibliographic database of section 2.1.2, the database can be shown in a directed-labelled graph as in Figure 2-2. It is noticeable that new nodes of root and article have been added to the database to satisfy a hierarchical design.

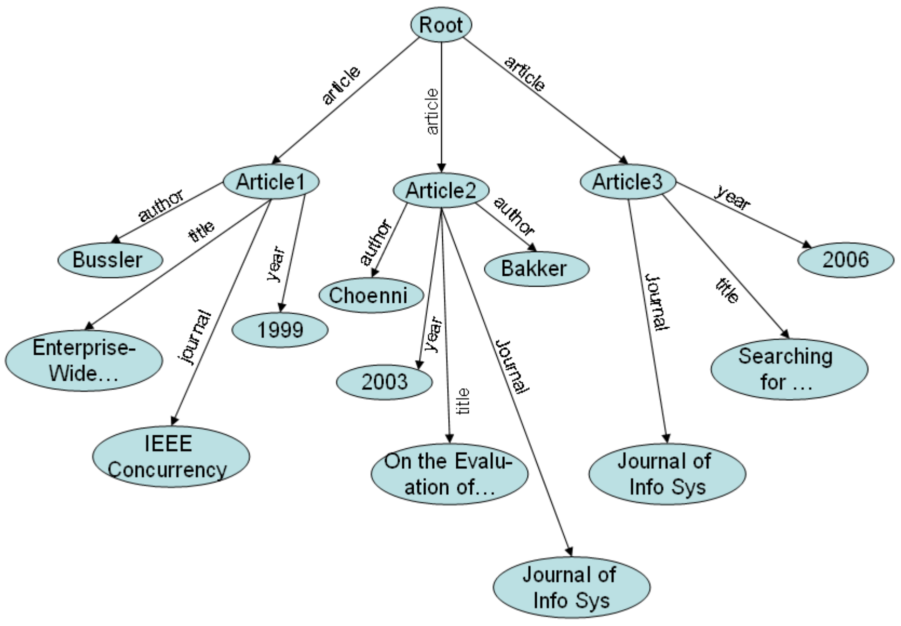


Figure 2-2: The sample database in a directed-labelled graph

The illustrated graph then can be used to build an XML listing of the sample database, as listed in Figure 2-3.

XML supports using ID, IDREF pairs, which can be used to modify the listed XML in order to reduce redundancies, if necessary. This is equivalent to changing the graph to have some multi-input nodes. In addition, a schema written in XMLSchema can be used to validate it, as listed in Figure 2-4.

```
<root>
    <article>
        <author>C. Bussler</author>
        <title>Enterprise-Wide Workflow Management</title>
        <journal>IEEE Concurrency</journal>
        <year>1999</year>
    </article>
    <article>
        <author>S. Choenni</author>
        <author>R. Bakker</author>
        <title>On the Evaluation of Workflow Systems in Business</title>
        <journal> Journal of Information Systems Evaluation</journal>
        <year>2003</year>
    </article>
    <article>
        <title>Searching for e-Business Performance Measurement systems</title>
        <journal> Journal of Information Systems Evaluation</journal>
        <year>2006</year>
    </article>
</root>
```

Figure 2-3: XML listing of the sample database

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="root">
        <xs:complexType>
            <xs:all>
            <xs:element name="article">
            <xs:complexType>
            <xs:all>
            <xs:element name="author" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="title"/>
            <xs:element name="journal"/>
            <xs:element name="year"/>
            </xs:all>
            </xs:complexType>
            </xs:element>
            </xs:all>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 2-4: XMLSchema listing of the sample database

2.2.3 XML: A Ternary Approach

XML can be thought as a series of ternary links between elements, attributes and textual values. In an XML tree, there is a hierarchy of elements. The XML hierarchy can be flattened to form a ternary node-link structure. It is noticeable that converting an XML listing to a series of directed labelled graphs is a very common way of representing, and storing XML information [175]. The directed labelled graph can be thought as a demoted version of the general ternary node-link structure, because nodes denoting labels can not be reused in any other link.

It isn't necessary to use this labelled graph conversion method, instead a more general approach may be used to convert an XML listing to a complete ternary node-link structure: Each sub-element adds a branch to this hierarchy and connects its body contents to its super-element. So an element can be considered to be an association between its super element and its body. In the same sense, an attribute name is an association between an element and its textual (or referable) contents. This will be described more in section 0.

2.3 ZigZag

ZigZag is an information paradigm that has been developed by Ted Nelson over the last decade [117, 131]. ZigZag is particularly suited to representing scientific or other real-world information that can be difficult to model using paradigms developed for "man made" information [121, 122]. Many information models – in particular relational databases – were developed primarily for business use. Such information can fit neatly into rows and columns, and it is generally possible to modify business procedures so that the information fits such structures. This is not always the case with information from the real world – such as scientific information. It is not possible to change the structure of a protein or the path of a river so that they fit in with the information paradigms intended for other application. However, the fluid schema-less structures of ZigZag allow such structures to be modelled easily.

In ZigZag, cells are atomic information units that can be interconnected with directed links along dimensions – which may have meaning or may be arbitrary. A cell cannot have more than an originating link or terminating link along any given single dimension (i.e. one in, one out). All cells may exis in all dimensions, although they may or may not be connected to anything. When a sequence of cells is connected along a particular dimension, that structure is called a "rank". Cells may be transcluded so that they appear in any rank in which they may be required. These simple rules provide a multi-dimensional space which can be as intricate and complex as is required by the information that it represents. A detailed data structure based on ZigZag, called *zzstructure*, has been introduced by Nelson in [129], with primitives named *zzcell*, *zzlink* and *zzdim*.

The linked-list of software engineering may be represented by a one-dimensional zzstructure and a spreadsheet by a two-dimensional one, although these are simple examples. The number of dimensions is not in any way limited, and in practice many dimensions are often required for representing real-world information. Another important feature is that loops may be constructed by linking two ends of a rank – something that is not possible in a spreadsheet, or many other information structures.

A great advantage of such an information modelling is that a zzstructure has no schema and so the information can be "grown" easily without the difficulty of structural change, as opposed to the relational database. Adding new dimensions, cells, and connecting cells along dimensions are the only functionalities that are needed to grow the information space. Moreover, zzstructures are built using ZigZag itself – dimensions, links and transclusions being themselves stored as cells in the system.

Nelson's vision of ZigZag includes a user interface as well as an information model [129]. However, it is possible to abstract the user interface from the information model, as will be shown in section 3.3. The multi-dimensional zzstructure may be viewed in either a two or thee dimensional space, using a variety of visualisation techniques. The user may then traverse this space by moving a cursor along x, y and sometimes z axes which may represent any of the dimensions of the underlying zzstructure. In order to make zzstructures independent of any external description framework, ZigZag itself is built almost entirely of self-descriptive zzstructures – in a "turtles all the way down" philosophy – as quoted by Stephen Hawkings [95]. For example, dimension names are stored in cells of a particular rank (called d.dim)– adding a new dimension is simply a question of adding a new cell to that rank.

There are various implementations of ZigZag, which provide different approaches to visualisation. The best known implementation is GZZ [110], which has interchangeable modular "views" – each of these determining how two or three user-selected dimensions are rendered in a pseudo-3D interface.

The way that the multi-dimensional zzstructure can be viewed as a two dimensional space (like on the monitor screen) depends on how the user selects visual x and y dimensions. The user can traverse between cells in that space by moving a *cursor* along x and y visual dimensions. Also there are different possible ways of mapping zzstructure in a two dimensional space. All of these functionalities has been realized by designing GZigZag, a Java-based zzstructure platform [110].

2.3.1 The Main Challenges

Transclusion (called cloning in ZigZag) is an important and challenging property of zzstructure. This allows a single cell to appear in any number of different ranks. It is obvious from ZigZag definition that a single cell can be involved in various links along various dimensions, and there is no need to define it repeatedly. The question is how to differentiate between each existence of a single cell in different ranks? There should be a mechanism to individually refer to each of those existences. Also there should be a mechanism to visualize a single cell participating in two different ranks if both are being visualized in a two dimensional space. Because of these reasons, zzstructure allows defining a zzcell once, and use its *clones* in different positions. Clones represent the main zzcell with all of its associated data, but with different references. The cloned zzcells are different cells in zzstructure, and they are connected to each other along a special dimension called *d.clone*. The main cell is positioned at the head of such a resulted rank. Clones appear in the zzstructure as though they are separate cells, although there is actually only one cell being represented in many different contexts.

There are two other challenging issues in data manipulation in ZigZag, both about how to map information on ZigZag topological principles.

The first issue is the main restriction of ZigZag on having a single right-hand and a single left-hand cell along a single dimension, which consequently makes the use of transclusion necessary. ZigZag has no mechanism for a establishing direct one-to-many relationships, so that relationship must be broken to a number of one-to-one relationships to the cloned cells, which has its own disadvantage of resource waste. Alternatively, the "head-cell mechanism" can be used. In head-cell mechanism, a rank of cells is interpreted as a one-to-many relationship between the first cell (head) as one side, and the rest of cells as the other side of the relationship. This is the same mechanism that has been used in zzstructure to relate a cell to all of its clones along a dimension called d.clone. The disadvantage of head-cell mechanism is that it is based on a "constitution" not on the topology. For instance, changing the order of cells along a dimension (except for the head cell) has a deep topological meaning on data, but has no meaning in establishing a one-to-many relationship in head-cell mechanism. Other similar alternatives are also possible, like making a rank of all related cells from many-side and linking the one-side cell to the head of that rank along a second dimension. This has also the same disadvantage and ambiguity as the head-cell mechanism.

To address the mentioned problem of one-to-many relationship in ZigZag, a detailed discussion has been provided in Appendix B. The main idea in that appendix is to keep the freedom of data manipulation in ZigZag, while not entering the Cartesian environment of the relational database, and still have the choice of establishing direct one-to-many relationships. The proposed solution consists of ZigZag elements plus another concept which is called Macro-cells. The result is no longer ZigZag.

The second issue is the ambiguity of basic ZigZag elements. There is a constant need for additional information or constitutions in order to understand the real meaning of a topology in ZigZag. The mentioned ambiguity of using head-cell mechanism to express one-to-many relationship is an example. More generally, if A, B and C are forming a rank, the meaning of the relationship between A and B is not always same for B and C. Only a subset of dimensions can have clear meaning when joining more than two cells, depending on the concept of linking. For instance, the dimension of *d.age* can not be used to link more than two cells, while the dimension of *d.son* may accept more. The other ambiguity is the meaning of a reverse links in ZigZag. Even the meaning of direct link is not always clear. For example, while the dimension of *d.son* can implicitly express the meaning of *d.father*, there is no mechanism to relate these two reverse meanings to each other.

2.3.2 An Example

Recalling the bibliographic database of section 2.1.2, the illustration of that database in ZigZag will be provided here. Illustrating a zzstructure on paper is difficult because ZigZag is intrinsically multi-dimensional and does not sit well with the two dimensions of paper. However, using the same visualization techniques as used in GZZ, Figure 2-5 shows how that database might be represented in zzstructure. Here, three articles are linked as a rank along the d.article dimension. Each article has a title linked to it along the d.title dimension. There is no need for cloning here, because each title is unique and each article only has one title. Each article is linked to each author along the d.author dimension, but where there is more than one author the article is cloned (there is still only one article, but it is represented in the context of each author). When an article is cloned it is linked in a rank along the d.clone dimension. The journal in which the article is published is linked to each article along the d.journal dimension, being cloned as necessary. Each journal is unique (there is only one "Journal of Info Sys"), but each journal publishes many articles, so in this zzstructure the journal is represented in the context of each paper, and again linked in a rank along d.clone.

It is important to imagine all of the 2 or 3 dimensional views as different looks on a same space. Also because cloning needs a separate dimension (d.clone), it has always been selected as z-dimension (clones are shown by dotted lines). For consistency, two dimensional views are used if no cloning was necessary.

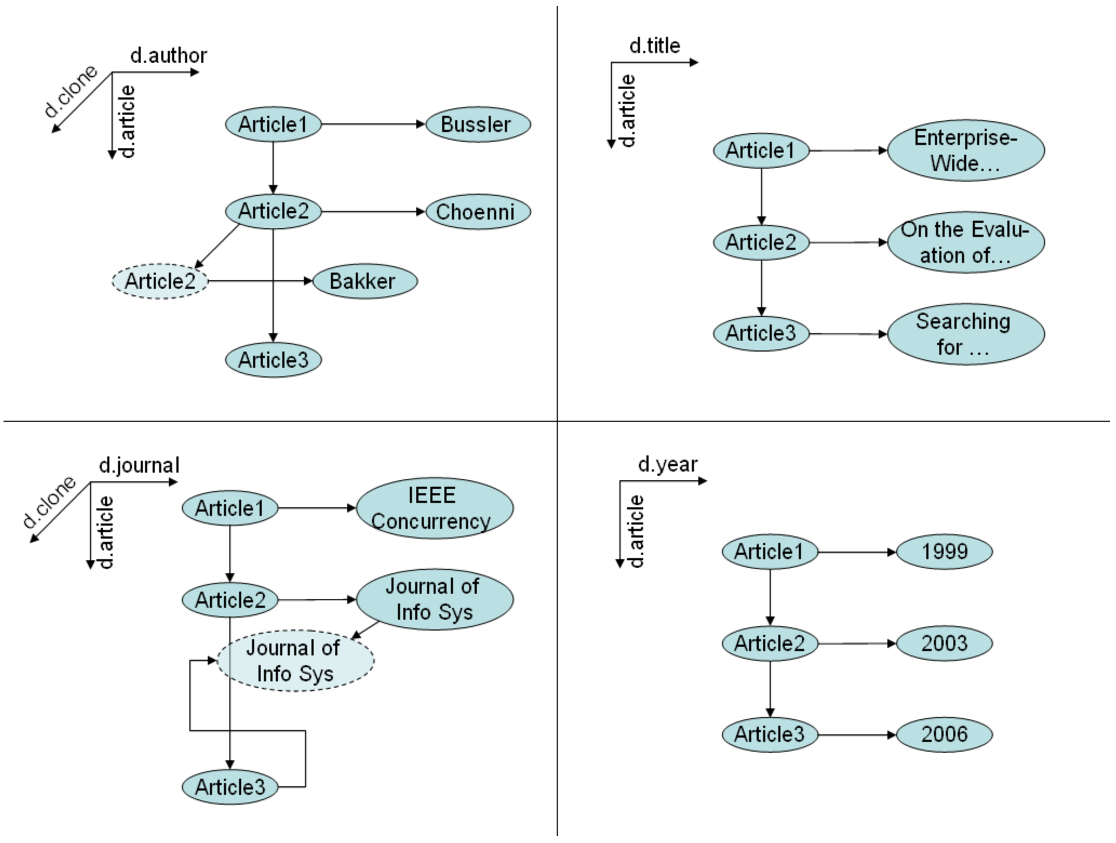


Figure 2-5: The sample database in Zzstructure

2.3.3 ZigZag: A Ternary Approach

Dimensions are the key elements in ZigZag that are utilised to express and aggregate links between nodes. Thus the structure has high informational strength, and associative meanings are significant. Since links in ZigZag are meaningless without specifying the dimension, ZigZag may be easily expressed in terms of ternary relationships: A zzstructure may be reduced to a set of triples. Each triple consists of an originating cell, a dimension, and a terminating cell. All cells and dimensions must then be defined in these terms. Thus a link in a zzstructure is set of three nodes: the left-node, a dimension and the right-node. Using this approach, it is possible to provide a formal definition of ZigZag as follows:

ZigZag is a triple of (C,D,Z) where:
C: Set of all cells
D: Set of all dimensions
$Z \subset C \times D \times C$
For each $(x1,d1,y1) \in Z$, $(x2,d2,y2) \in Z$

 If $x1=x2$ *and* $d1=d2$ *then* $y1=y2$

 If $y1=y2$ *and* $d1=d2$ *then* $x1=x2$

The last two line of this definition guarantees the uniqueness of linking cells along a single dimension. The first condition checks any two triples, to see whether they have the same originating cell and the same dimension, and if so, then the terminating cell must also be the same. Similarly, the second condition checks any two triples, to see whether they have the same terminating cell and same dimension, in which case the originating cell must then be same. Since there are no repeated members in a set like Z, being same in three elements means that two triples are in fact one single triple.

Zzstructure allows us to have an integrated approach to dimensions, because they are in fact stored in cells. By this view, if cells in zzstructure can represent dimensions, then the above definition can be changed to:

Zzstructure is a pair of (C, Z) where:
C: Set of all zzcells
$Z \subset C^3$
For each $(x1,d1,y1) \in Z$, $(x2,d2,y2) \in Z$

 If $x1=x2$ *and* $d1=d2$ *then* $y1=y2$

 If $y1=y2$ *and* $d1=d2$ *then* $x1=x2$

This is not the first attempt to define zzstructure in formal terms. McGuffin et al. in [116] and [117] have used various approaches to define zzstructure using graph-oriented, list-oriented and space-oriented techniques. The ternary definition of zzstructure that has been provided above is set-oriented, compared to the graph-oriented definition provided in [116].

There are other issues about this ternary approach to ZigZag paradigm that must be studied after defining the TRM. These issues will be studied in section 0.

2.4 Hypertext Models and Navigation

An almost universal feature of different definitions of hypertext is the non-linearity of structure [21]. The non-linear characteristic of both reading and writing in hypertext was a part of Nelson's original definition [128, 130]. Various definitions of hypertext are reviewed in [20], and the one used here is *"an interconnected structure of information which can provide non-linearity in reading and writing"*. Another definition that can be helpful in the direction of this thesis is *"Hypertext is the authoring and use, by people or machines, of associative relationships among information nodes."* [88]. This clearly defines hypertext in the context of node-relationship structures. In this thesis, the terms "Hypertext" and "Hypermedia" are considered to be interchangeable because the type of media (textual, visual etc.) doesn't make any difference in relating them together in the context of this thesis.

When the interconnectivity of information is considered (like in hypertext systems) *nodes* are the separated and abstracted pieces of information that can be interconnected in order to express another pieces of information. That's why in the context of the TRM, as will be described in section 3.1, a relation is itself a node.

In the context of hypertext systems, the working definition of link is also similar to that described in [20] as a *"trigger plus the retrieval action performed when that trigger is activated"*. In this thesis a link is considered to be a hypertext feature which provides the functional use of the interconnection.

The associative meaning of a link, or simply the *association*, is defined here to denote "any implicit or explicit relationship between two nodes of information". This may or may not denote semantics, depending upon the context.

According to the mentioned definition of hypertext, node and link, hypertext has the potential of expressing knowledge through a node-link structure. This structure has been modelled from several aspects, because hypertext systems have two human and technical sides, each one having authoring and reading sides. Reviewing the developed hypertext models is out of the scope of this thesis. Instead, some of the developed linking methods are studied in this section, which may or may not be a part of a general hypertext model.

Although the link structure looks clear in meaning, it is interesting to know that there is still no general "link model" for hypertext systems, which is an information model to manage links, as mentioned by Carole Goble in her keynote address to the HT'07 [88]. Also she considers the main missing part of such a link model to be a link navigation model, which models the way users navigate between nodes of information. According to the definition of hypertext used above, this navigation model shall ease a non-linear reading and writing through a node-link structure. Also because hypertext is defined to be potentially a knowledge system, a well-modelled navigation shall provide a good media for transferring knowledge between authors and readers *through navigation*.

In the rest of this section, the concept of *Knowledge-oriented Hypertext* will be explained in section 2.4.1. Then in sections 2.4.2 to 2.4.5 some related approaches to hypertext links will be studied, varying from implicit modelling of "association" to explicit ones. Finally in section 2.4.6 the studied related works are reviewed again from a new point of view in order to justify it in the main direction of the thesis.

2.4.1 Knowledge-Oriented Hypertext

Based on the provided definition of hypertext and knowledge, a hypertext system has a potential of being a knowledge transfer system, because hypertext and knowledge are both based on a node-relationship structure. Here the term "Knowledge-oriented hypertext systems" is defined as those that are designed to provide an optimum knowledge transfer while being traversed. Most studies of such systems have focused either upon the use of hypertext functionalities to build a knowledge system, or through the incorporation of knowledge system characteristics into an established hypertext. In this thesis, these two areas are not considered to be separate, but are instead realized as a single system.

If the link structure is a key place for accommodating knowledge in hypertext, then the meanings of association between the source and destination of each link have the highest importance in knowledge-orientation. This is because without clear associative meaning in navigation, discerning the intention of the author, extracting what information items the system contains and determining how they are connected puts far too much interpretation responsibility, and hence cognitive load upon the end user [134].

Thus knowledge-oriented navigation is not only concerned with accommodating traversal of the links but also with the discovery of associations [17]. Another interesting approach comes from [17] where "links do not express meaning by themselves, but express meaning through their navigation. It is not in he links themselves, but by navigating through the links that the meaning of the links becomes clear". This highlights that the total behaviour of a set of links when navigated by the user forms the knowledge transfer environment.

Some other works such as [16] use the term "Intelligent Hypertext" for the same concept, but it is avoided in this thesis because of the definition of knowledge used herein. Knowledge-oriented hypertext can still have all the properties of the other types of hypertext system (can be called *information-oriented hypertext*), if it is not intended to be used as a knowledge carrier. Another alternative term is Knowledge-based Hypertext in [17], which is "one that is able to explicitly represent and actively manipulate the semantics of its informational contents".

In order to reach a knowledge-oriented hypertext, many approaches have been taken by researchers, varying from overlaying solutions to fundamental model changes. Arents and Bogaerts in [16] count two distinctive groups of approaches. In the first, knowledge is expressed within the hypertext network itself; and in the second knowledge is expressed on top of the hypertext network, as a separate layer. They mention that the majority of works lie within the first group, indicating that this is due to the similarity and potential of basic hypertext model elements to be utilised in knowledge systems. However, the second group offers greater promise in integrating knowledge into hypertext systems. They provide a hypertext design model called Model-Map-View-Praxis or MMVP architecture to support this idea (more details in section 2.4.3). A related two-layer approach integrates hypertext into the design of a knowledge-based environment, as exemplified by JANUS [76], in which the construction tasks of the system are supported by graphs, and argumentation is supported by a hypertext system. Nevertheless, the above separated approaches have many overlapping features: for example, the addition of knowledge-links to the standard extant links of a hypertext system (exemplified in [44, 89]) acts as an overlying logical layer, while still utilising an established hypertext network.

2.4.2 Links with No or Implicit Association

Putting association implicitly in content, or even using no association, is the most common approach to hypertext linking, as exemplified by HTML [183]on the World Wide Web. Thus in order to effectively navigate a link, a conceptual understanding of the relationship must be maintained in the user's mind [172]. This can be facilitated by the HTML anchor text, by the mental model associated to a graphical icon, or even by a previous experience of the user in Web navigation. Thus according to the definition of the knowledge-oriented hypertext, HTML links show less clarity for the association than expected.

Spatial Hypertext [167] is an intermediate solution where the association is implicit in link structure visualization (i.e. a spatial view illustrates an instant logic of relationships between the nodes). Indeed, any overall strategy in organizing links, such as organizing them into a hierarchy, is also likely to help the user to effectively visualize the system.

In WebML (Web Modelling Language) [46], a set of descriptive model, an XML-based language and graphical notations is developed for conceptual designing aspects of the Web pages. The WebML descriptive hypertext model includes some sub-models: Composition, Navigation, Presentation and Personalization. In the Navigation sub-model, the links are defined as two types of "non-contextual" (when they connect semantically independent node) and "contextual" (when the context of the destination node of the link depends on the context of the source node). This categorisation is almost referring to the presence or absence of what is called "association" in this thesis. WebML is developed for designing the Web sites (and specially for knowledge systems on the Web [73]) and thus may not be used for other non-Web hypermedia systems. In terms of links, it is limited to the Web style of linking and may not be used for dynamic links modelling. The association in this model is an embedded and implicit element of the WebML links.

The implicit associations (or no association) approaches above do have potential problems. Firstly, It is possible, indeed quite likely, that in many cases the resulting hypertext does not engender the same conceptual model on the part of the reader that was intended by the author [125]. Furthermore, in most cases these kind of links are more suitable for free navigation of information rather than efficient knowledge purposes or semantic retrieval of content [106].

2.4.3 Rank-Promoted Links

In the object layer of many classical hypertext models, links are second class objects (as either in the older models like Dexter Model [92] or in the recent ones like Hypermedia General Meta-model [168]) or dependant objects (like in some of the representations in the Binary Relation Model of links [19]). However there exist completely contrasting approaches, where the rank of link objects is promoted either by their primacy over nodes or alternatively by handling nodes and links as equal-rank objects. Two related approaches will be studied here.

In order to emphasise the structure of hypertext systems, the philosophy of "Structural Computing" has been introduced in [135, 136], where the links are considered as first class objects in basic design and relationships as the atomic building blocks. This is similar to seeing a graph as edge-based rather than node-based. Moreover, a hypertext system is considered as just a special case of this general philosophy. A hypertext implementation based on the Structural Computing is IUHM (Information Unit Hypertext Model) [127].

There is also another approach towards upgrading a link's order from being a second class object. In the Model-Map-View-Praxis (MMVP) [17], explicit knowledge manipulation for both nodes and links is emphasized. The idea behind seeing nodes and links as equal rank objects is that in MMVP, nodes and links are two abstract objects in lower layers that must be instantiated for representation using the upper layers. A link has a navigational behaviour which can be used in some similar places, and not simply as a reference to the source and destination anchors. Nodes and links are not explicitly stored in the lowest (Model) layer of the architecture, but are implicitly extracted from the information units and the information semantics in that layer.

2.4.4 Typed Links

One commonality in various hypertext models (like in [91, 155]), is the view that a link has at least two essential data fields: source and destination. Having only these two fields cannot express any kind of intrinsic knowledge, since there is no inclusion of meaning for the relationship between the source and the destination, and this is analogous to a sentence without a verb. The enrichment of links by semantic meanings has been called as semantic linking by some authors [16, 106, 126, 167]. Indeed, the existence of semantically typed links has been counted as one of the main evaluation factors in navigational model design of hypertext systems [54], an infrastructural component in "third-order hypertext systems"

[17] and as a new added feature to "the fourth generation of hypertext systems" [32]. Enriching the nodes and links by semantics is also known to be the underlying step towards converting the World Wide Web to the "World Wide Knowledge Web" to provide semantic filtering/visualization of the Web pages from different perspectives [40].

Adding an explicit type attribute to the link feature can explicitly contain the link association, or the way two or more nodes are related [54]. The added link type is metadata to describe or enhance the usefulness of data [44]. Also link typing researches are not always directed to providing semantic links, and they can be for descriptive or more general purposes. In the following paragraphs, some related issues and works on link typing will be studied.

HTML 3.2 [182] and later versions support generic type linking: CLASS, REL and REV attributes of <A> and <LINK> tags have been designed in order to handle link types. Unfortunately this has been rarely used and most of the known Web browsers ignore them [32, 40]. It is noticeable that they have been used by stylesheets to change the look of the Web pages, diametrically opposed to their original intention. This facility can help users and computers to understand various link categorizations, either in terms of their semantic or other purposes. REL and REV attributes are used in a bi-directional manner for navigational sequencing of web pages, creating structural hierarchy within web pages, and for some special purposes such as defining author, copyright, etc. REL and REV accepts pre-defined values but CLASS attribute accepts free text for further description of the link [32].

Whether or not Web browsers can use the built-in HTML link typing for presentation, and whether or not these types are for semantic purposes, HTML link types have their own advantages to help Web searchers (like search engines or Web agents) when HTML sources of Web pages are processed independently. This process can help finding related Web resources more easily and intelligently. As an additional advantage, some other processes can use the link types to analyze and rank the Web pages based on incoming links from the other Web pages. A more detailed study about these kinds of search methods has been done in [153]. The disadvantages of HTML link typing in supporting knowledge-orientation include non-semantic provisional design, the lack of standard presentational support by browsers and finally limited types for bidirectional semantics.

Also an explicit and formal manner of relationship, named *Metalevel links*, has been proposed in [172] to address the problem of informality in usual HTML links. The meaning of a link has been added as a type attribute to the link data model. An implementation of this idea is *WIS* [172], which has the following other advantages over conventional websites: bi-directional linking, different views based upon filtering of the link types, intelligent searching based on relationship types, provision of a platform for implementing workflow systems, and distributed and open architecture. As an example, a link type can be expressed as "parent-child" which can be used for an intelligent search based on "sibling".

A similar approach is used by Oinas-Kukkonen, in which link types and link keywords have been purposed to address another important problem: *Many complain they do not know where a link will take them?* [141]. By knowing the link types, users may have a better way of knowing the target prior to navigation. Such links have been named as *rich links* in [141] with some improvement on the system's efficiency: preserving the information context for the user in addition to better information organization, and benefits in collaborative design. Although Oinas-Kukkonen has no suggestion on how to implement such links, a similar implementation has been described in [193] in which a link can have multiple destinations, distinctive by several link types and the user selects one of the link types from some appearing pop-up menus before the link activation.

Another approach has been taken in the *Trellis Model* of hypertext [169]. This model is based on *Petri-net* (a widely known workflow analyzing scheme explained in section 6.1.1), and tries to benefit from the existing analyzing algorithms which have been developed on Petri-nets. In this model, the fact that a *transition* object intermediates each two *place*s in a Petri-net, is mapped into the fact that a link intermediates two nodes of information in the hypertext systems. Also the *firing* process of Petri-nets which transmits *tokens* between places is mapped into navigation which transmits control between documents. Link typing can take place by this analogy, because links have as many attributes as transitions, including type. The model is not a design model, but a functional model of hypertext and this type attribute has only instant browsing meanings, which may or may not have associative meanings [81, 170].

Another approach is *RMM* (Relationship Management Methodology) [100] which is a framework for object orientated hypertext design. Although in its underlying data model

(*RMDM*) an entity-relationship (E-R) diagram deals with user navigation of hypertext rather than its design, RMDM differentiates between navigational and associative links. The links in the E-R diagram are meaningfully labelled for both types of links in the context of the object orientation. For example, a link between a course and a teacher's name can be labelled as "taught by". In the final user interface design, these labels will themselves be link anchors inside the documents, i.e. at the end of a page which contains the specification of the course. In this view, the links have explicit meanings, but are not extracted from the information nodes.

2.4.5 Links in Open Hypertext Approaches

The term "Open Hypertext" has been firstly used in 1989 by Sun's Link Service project [144]. The openness mainly refers to the free access of different applications to "Link Server" as well as documents, so each application can integrate document linking services into their standard functionalities. Then the researchers in the University of Southampton, have worked on various aspects of the "Open Hypermedia Model" and exemplifying the idea by developing Microcosm Link Service [63, 79, 93, 186, 187].

In Microcosm, the user reacts with some "viewers" which can be any document displaying application. The heart of Microcosm is a document control system which controls the passage of "messages" between the viewers, linkbases and "filters". Each of the filters can then block or change the message before passing it on. For example if a link source in a document is selected by the user, the message of requesting the destination may be passed to the linkbase through the filters and be responded by another message.

In the open approaches to hypertext, links are logically kept out of the contents of documents, in some "link databases" or simply "linkbases" [63]. Using linkbases can provide more flexibility in managing the link structure. For example, the "linkbases" can be updated, computed, added or adapted independently from the content, as well as utilizing some automatic linking algorithms [63]. Also various linkbases are attachable to a single document and a linkbase can serve different documents. It also has the benefit of more efficient handling of large and numerous documents compared to the embedded-links -or closed- approaches.

In terms of the link structure, managing links in some separated linkbases, allows us to have as many explicit modelling elements as necessary for each link, regardless of the contents that the link is going to appear in. Thus the open hypertext approaches may provide opportunities for applying explicit association elements to the hyperlinks.

In the methods of adding computed links and automatic linking, automatic processes attempt to enrich the informational structure by constructing new links. This approach is very much related to the building of a knowledge-oriented hypertext because the automated embellishments of the structure harness external sources of knowledge (e.g. in [20]). However, this method may not change the link structural model and the added links may still have implicit meanings to the user. Consequently, a lack of direct knowledge transfer from the author to the reader may still exist.

Also in COHSE (Conceptual Open Hypertext ServicE) [44, 89], conceptual metadata about hypertext documents is used to add pre-computed links to the pre-existing navigational links. The link generator uses several software modules to recognize potential anchor points such as ontology services, other external linkbases, RDF repositories of the Semantic Web, or some explicit metadata descriptions inside the documents, such as the <META> tag of HTML. COHSE can convert a set of conceptually-unlinked documents like normal web pages to another set of linked documents. A particular useful application of this appears when a single document can be enriched by several types of knowledge, each for a specific group of readers.

Another approach is taken by Bieber in [29, 30] by introducing DHE (Dynamic Hypertext Engine) as a method of automatic links addition to hypertext, based on the analysis of existing relational databases. These databases are actually sources of supportive knowledge and the created links are enrichments of hypertext by those knowledge sources. The applied analyzing algorithm (RNA: Relationship Navigation Analysis) is based on the internal joins of the relational databases.

There have been many other descriptions of adding computed links, (e.g. [12, 31, 51, 203, 204]), all of which attempt to add computed knowledge-supported hyperlinks over the pre-existing ones.

Also FOHM (Fundamental Open Hypertext Model) [64, 119] was proposed as a single framework for modelling interoperability between several open hypertext standards. Because of the generality of FOHM, the "association" has been incorporated as a modelling element. The set of associations is defined by the Cartesian product of three sets of *binding vectors*, *relation types* and *structural types*. Relation type is itself Cartesian product of a set of *names* and a set of *features spaces* while the latter is a set of all possible properties that must be defined in each binding of an association. The relation type has no direct involvement in semantic linking, as it has more functional involvement, distinguishing between different behaviours when the link is traversed.

Another method to use the Web infrastructure as an open hypermedia system is XLink. XLink is the W3C recommended method to incorporate links in XML documents [185]. XLink are special elements within XML documents that can represent unidirectional links between two other XML elements. In addition to the simple one-to-one links, XLink support "extended links", in which elements can be related in one-to-many or many-to-many manner. It is noticeable that XLink does not itself produce hyperlinks but uses elements of a special namespace (XLink namespace) to notify a reader application about the existence of some links. So it is absolutely due to the reader application how to react to the XLink elements in an XML document. XLink has been considered as a method to use the Web infrastructure as an open hypertext system by greater abstraction of links from nodes [15, 106]. By using XLink, each link can have more structured attributes for linkage. The attributes of the links which are defined neither in the source nor in the destination, are a good opportunity to store the link semantic and/or types. The main linking element in XLink is <bind> which has attributes including "from", "to", "type" and "role". The last two are where the associative elements of a link can be stored.

However, at the time of writing this thesis, XLink 1.0 (2001) was the only finalized version of XLink recommended by W3C and one of the main current issues with XLink is the lack of implementation support by the Web browsers. Only the recent versions of Mozila Firefox and Netscape have a very limited support for "simple" links and no major Web browser supports "extended" links.

Also Frei and Stieger in [80] have defined a hypertext link to be consisting of four components: <*t, i, s, d*>, where *t* is the link type, *i* is a set of link attributes, *s* and *d* are source and destination node of the link. *t* is not itself a semantic type of the link, but rather a flag that specifies whether the link is of type referential or semantic or at most distinguishes several subtypes of semantic links. They mention that *the intention is to restrict ourselves to a few link types so that their semantics may be understood fully by authors and users*, so it is clear that this link typing relies on the user's mind to interpret the exact meanings of the links.

2.4.6 Hypertext Links: A Ternary Approach

After reviewing the related hypertext linking approaches, it is observable that the "association" elements have been considered in several ways, either implicitly or explicitly. As a summary, the way that the "association" has been modelled can be categorized as follows:

1. *Association implicit in structure:* Where clear illustration of structure can transfer an understanding of the meaning of each link (as in Spatial Hypertext [167]).

2. *Association implicit in the source node:* Where observation of the hyperlink together with the help of reader's mental model may express the meaning of links (like in Web's usual links) [172] or even when some separated link anchors in the source node express the association (like in RMM [100]).

3. *Association explicit in structure:* Where the storage of a link includes some information about its meaning (like in XLink [15]).

4. *Association explicit in other nodes:* Where the association may be explicitly stored in a separated node. None of the studied related works can be explicitly categorized under this category. However, the Structural Computing is the nearest one to the case of "association explicit in other nodes". There is a possible ternary approach to the concept of the structural computing inspired from Nürnberg's work in [136][1].

Implicit methods of link associations have the significant advantages of simplicity and no storage overhead, but are less desirable from the perspective of knowledge-orientation, as

[1] In structural computing, the information tends to be stored primarily in structure and secondly in nodes. The "structural atoms" or *bundles* have been introduced in as the first class objects. Data elements of each bundle include a set of *ends* and a set of adjacent bundles per each end. This can be partially illustrated by looking at an edge of a graph as the main object which has some ends (nodes), and the next edges are its adjacent through its ends. The resulted model (called EAD: Elucidate; Analogize; and Delete) allows bundles with more than two ends, which is impossible to illustrate in normal graphs. Instead, Nürnberg propose an alternative bipartite graph in which the nodes are of two types A and B and the edges can only link nodes of different types. Then EAD's bundles can be seen as A-nodes and EAD's ends as B-nodes. Then a real bundle in EAD is in fact two adjacent edges (ABA or BAB) of this graph, when different adjacency selection means having multiple ends. The multi-end nature of the bundles in structural computing is implemented by triples of (end, bundle, end) or (bundle, end, bundle). As a conclusion, two-ended bundle are following a binary approach while ternary bundles are following a ternary approach.

they rely heavily on the reader's mental model. Where associations are explicit in the structure, they are stored in the form of metadata about each link. However, this information is not necessarily transferred to the user's mind through link navigation. Hence the advantage of this type of linking lies more in the efficiency of search methods and information retrieval that it offers. Sometimes link attributes are used for other purposes rather than associative meanings (as in FOHM [119]), and they are usually too restrictive in size to store a complete associative meaning/description of a link. By contrast, explicit associations in the source node are more suitable for knowledge transfer and for directing the user's mind, although they may not be as efficient as storing them explicitly in structure for intelligent information retrieval purposes. Furthermore, the anchors are often too short to express full associative meanings (or if this is not the case then the readability of the source document is likely to be impaired).

If associations are stored explicitly in a third node, then there is higher information overhead for each link, in comparison with all of the other methods described. However, there are a number of distinct advantages to this approach.

Associations stored explicitly in third nodes can express the available information about the link to the greatest possible extent, because all such information is consistently stored in other nodes. This method promotes associations from being attributes to full navigational information, because in this approach, the association is one of three basic elements of a link, with the same rank (sitting alongside source and destination). The "attribute" view to the link associations has caused their exclusion from incorporation into various navigational models ([19, 20]). Lastly, this method supports the openness of the hypertext systems (where openness is defined earlier in section 2.4.5) because it includes management of many link specifications outside of the link structure. As such, a link structure in the resulting open hypertext system has only pointers to the real information concerning the links and these informational items themselves also being stored as nodes. This is hence a higher abstraction of nodes and structure.

It is now observable that a complete link and navigation model shall model the associative element as explicitly as possible. If association is explicitly modelled in a third node, then all of the necessary information about the link is stored in that node, rather than somewhere in the source or in the destination.

2.5 BRM: Binary Relations Model

A fundamental work on navigational modelling is Ashman's work on the "Binary Relations Modelling" or "BRM" in [19, 20]. BRM can cover all of the possible binary navigation in a formal method. Although the BRM is a model for hypertext and thus it could be covered under section 2.4, because of its special importance as a formal predecessor for the TRM, it has been studied individually in this chapter. Although the BRM has been introduced in the context of hypertext, it may be studied in a wider context as an information model. After introducing the BRM in section 2.5.1, it will be reviewed by a ternary-based look in section 2.5.2.

2.5.1 BRM Link Model

The BRM [19, 20] is a way of enumerating all the possible ways of implementing link types in a hypertext system. It begins by identifying the salient features of binary relations from a hypertext point of view. This hypertext sensibility influenced the necessity of considering different representations, since the pure mathematical models of binary relations were not subject to real-world problems. For example, the volatility of the underlying set of elements in a relation, which in a hypertext and Web context, are manifested in implementation difficulties such as broken or disoriented links, and link completeness. The BRM abstracted out of real-world hypertext systems basic differences in the underlying link creation and maintenance processes, which are described in terms of the different representations within the BRM.

The BRM formulates all the possible ways of implementing link types in a hypertext system, by considering purely the navigation model, and focuses on general representation of binary relations regardless of their applications or visualizations.

The key features of the BRM are *endpoints*, *links* and *relations*:

1- An *endpoint* is any addressable "thing".

2- A *link* is a connection from an endpoint to another endpoint.

3- A (binary) *relation* R is a subset of S^2 (the Cartesian product of S upon itself) while the model space S is the set of all endpoints.

Then the BRM considers how relations are comprised, determining that there are three features:

1- *The source set* – elements which occur on the left of the relation, the "from" elements;

2- *The destination set* – elements on the right of the relation, the "to" elements;

3- *The incidences* – marking which of the sources is connected or related to which of the destinations.

Also it also considers how relations are utilised, primarily from a hypertext viewpoint, but with more general applicability. It does this through asking a series of "navigational" questions. Any arbitrary endpoint may be characterized by the following four main navigational questions:

1- *Source Existence: Is this node (x) the source of any link?*
$\exists (x,*) \in R$

2- *Destination Identification:* Where can I go from this node?
$\{y \in S \mid (x,y) \in R\}$

3- *Destination Existence:* Is this node (*y*) the destination of any link?
$\exists (*,y) \in R$

4- *Source Identification:* What nodes are linked to this node?
$\{y \in S \mid (x,y) \in R\}$

Questions 1 and 2 represent linking in the usual, "forward" direction, while 3 and 4 represent linking in the "backward" direction, so that bidirectional linking may be modelled. The answers to those four questions determine the various states of the sets of static or dynamic endpoints that are required to model the possible implementations of hypertext systems. These states are:

1- *Enumeration* – the explicit naming of all participating elements;

2- *Predicate* – the "filtering" of a set from a larger set by applying a set-membership selection test; and

3- *Expression* – a calculation (parameterised or not) that returns a set of elements.

An example of the enumeration state is a fixed set of journal titles on a webpage, which each one is a link source to its content. An example of the Predicate case is a function that determines whether or not the current user has access to the content of the journal. In that case, the link set membership is defined by a predicative function and the journal title is not a link unless that function returns "true". The example of the Expression case is when a function determines the destination of the link, e.g. the journal title can be a link to the abstract or to the full text, depending on the user's access level. The main difference between the Predicate and the Expression case is that the Predicate is a logical qualifying function, and the Expression is a function having a hypertext node as the output.

The predicate and expression states are also called "computed" , and the computation itself can be implemented in two different modes: pre-computed or dynamically computed [20, 22]. For pre-computed links, the link anchor in the source document is clearly specified after all the necessary computations, but in dynamically computed links, the eligibility of each node to be source or destination, is computed in run-time on user's request (like non-advertised links that are being advertised by hovering the mouse over them). This is reflected in the remaining questions that can be asked of a representation in the BRM, namely:

1- Link Existence: Is there a link between these two elements?

2- Source enumeration: What are all the possible sources of this set of links?

3- Destination enumeration: What are all the possible destinations of this set of links?

4- Link enumeration: What are all the links in this set?

The first of these is not a true navigation question because the identification of both source and destination endpoints is already known, the only question being asked is whether there is a corresponding pair of entry and exit points between them, i.e. "can one go from here to there?". The last three are not navigation questions involving decisions about if and where one can go from a given endpoint, but rather are queries about the whole set of links, whose results are independent of the reader's current position in the data collection. Pre-computation of all relation incidences (links) is the application of either a predicate or expression to calculate all the participants in any of the three constituent sets of a relation.

Having established how the defined sets form the relations, and how these may be represented, then a comprehensive enumeration of the representations for relations can be defined by considering all the possible combinations of possibilities for the sets making up a relation. To define that enumeration, one must also pay particular attention to how these representations occur in the real-world hypertext systems. These real-world observations support many of the theoretical observations, many being motivated by the challenges of maintaining valid hypertext links (equivalent to relation incidences) in a highly changeable information collection, such as the Web. This is a key limitation of those representations that use enumeration for any or all of their constituent sets, and the various representations of the BRM are discussed in terms of their ability to answer the navigational questions in a volatile and potentially infinite information collection.

One of the interesting possibilities that one may construct is *predicate-expression*, named *pE* hereafter, in which the source is nominated for being an endpoint by a computation (the non-advertised link source) then another computation takes the source and resolves the destination (either in pre-computed or dynamic fashion, as described in [20, 22]). This state, which is also called "Functional Links", is a generalization of all kinds of links in the BRM.

In section 5.8, more study on the pE state and the Functional Links will be done after introducing the TRM and the TRM-NAV. This will also consider a *Turing Completeness* approach to the BRM and the TRM (section 5.8).

2.5.2 The BRM: A Ternary Approach

According to the BRM's view, the semantics of the links are irrelevant to the relation model. The model is not affected by *why* any two endpoints are linked together, as its purpose is solely to characterize *how* they are linked. BRM excludes the link semantics (and more generally, associations) from being a navigational property of a link and leaves them as attributes. Also it has been reviewed that in knowledge-oriented navigation, users need to master the meaning of their navigational actions. Let us consider an open hypertext system with two different knowledge contexts (i.e. the associations are in two different domains). It is possible that two nodes are connected together in both contexts, but through different associations. Unlike their endpoint similarity, these two links have a significant difference when navigated in two knowledge domains.

In the context of knowledge-oriented hypertext, the importance of associations is too great to be ignored when characterizing a link. In addition, associations are not only attached to a navigation action, but also may have functional role. For example, a user may select an association choice after selecting a link source. Then the destination is dependent not only on the link source, but also on the selected association. It can be concluded that in knowledge orientation view on hypertext, some of the *why*s can be realized as *how*s in navigational modelling.

Furthermore, there are some areas in hypertext systems, in which navigational behaviour can not be covered completely by the BRM. It is predictable that these areas are where knowledge expressiveness is highlighted and/or when the structure of the system has more importance. The limitations of the BRM in covering such fields are because there is no independent characteristic for any relation incidence in the BRM [19].

Workflow Management Systems (described in section 2.7) are examples of when pure binary links are not able to serve user tasks an information system. Considering tasks of workflow as nodes of hypertext and its transactions as links, it is possible to build a workflow system over a hypertext system. The resulting workflow system has hypertext characteristics because it contains not only information about the definition of a process, but also provides non-linear navigation between its nodes. The navigation between nodes of a workflow is the ability of system to guide the user to go from one node to another depending on their decision from some offered choices. In this case, each navigation step consists of three parameters: source, decision and destination. This decision-based navigation has three navigation elements, which cannot be modelled by the BRM.

In the process of decision making, the user selects which type of processing they want to do on the current work case. Usually the decisions are source-dependent, i.e. the user selects their decision from a list of available choices, which are either predefined or computable to be available on the source node. However, there are possible source-independent (or enumerated) decisions, like suspension or jump.

Likewise, in the process of destination selection, the user and/or system determine the possible destination node(s). The destination can be more than one node, like distribution of a task amongst in-charged users. If the user selects a source-dependent decision the system determines the destinations (computed), otherwise the user selects the destination explicitly (enumerated).

Zigzag is another example of these BRM limitations, when BRM cannot model a cell-dimension-cell link of ZigZag.

The above issues can show a requirement for the extension of the BRM. This extension needs to take association into account as an independent node of information. Because the BRM may be viewed either as an abstract information model or a hypertext navigation model it may readily be extended to provide the TRM as an abstract information model (in chapter 3) and a navigation model (in chapter 5).

2.6 The Semantic Web and RDF

Many of the documents introducing the Semantic Web, start from this point that the current Web is designed to be human-readable, so why not make it computer-readable? And the motivation for this question is being expressed as scenarios telling about users who wish to do some specific logical queries but no software agent or search engine can satisfy them [14, 25-27], [142].

The basic idea is that the Semantic Web is not a new web, but an extension to it, by adding logical tags to the web objects, so the information is re-constructed in a machine-readable manner. This makes the web objects responsible for logical queries which come from some web agents or search engines, then the Web is searchable not only by its row contents, but also by its semantic interconnections [26]. The Semantic Web aims to build world-wide network of computer-readable semantics, instead of being human-readable [142]. It is interesting to know that the Semantic Web has been named as a hypertext, a knowledgebase and a database in different works [89].

Figure 6-2 shows the multi-layer architecture of the Semantic Web. The URI layer provides a global standard for referring to all the Web objects uniquely. The XML layer provides syntax, or basic language for describing information in all the upper layers. XML Query and XML Schema provide mechanisms to validate and access data written in XML. RDF (and RDF Schema) provides a data model (or language, or framework) for describing the

Web resources. It is used to write descriptive "statements" about each resource, using other resources, in the form of resource-property-value triples. The Ontology layer provides more mechanisms to logically enrich RDF-described data. OWL is the standard language used in the ontology layer of the Semantic Web (RDF and OWL will be studied more in the next section). Finally, the upper layers of the Semantic Web provide more AI mechanisms to make the web resources semantically reasonable, and the results of those reasoning reliable.

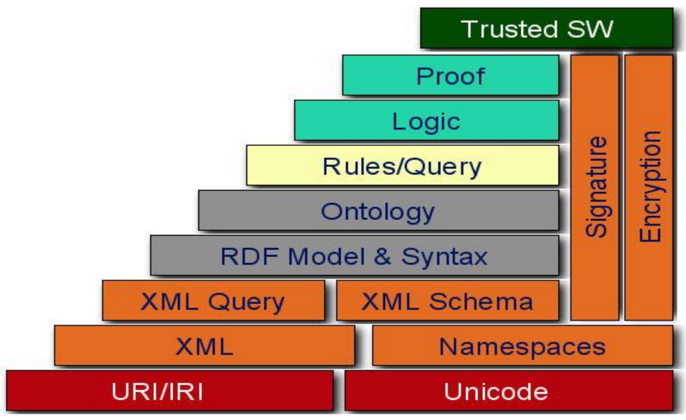


Figure 2-6: Multi-layered architecture of the Semantic Web
(from wikipedia.org)

2.6.1 RDF and OWL[1]

Among the layers of the Semantic Web, RDF and OWL are the core layers that make basic statements about resources. Since RDF uses triples for making such statements, it can be focused in this thesis. Although the Semantic Web is usually considered to be a subject in the context of hypertext but RDF can be studied in a wider context and be compared with the information model developed in this thesis. In this section, RDF and OWL elements will be briefly introduced.

The RDF framework includes two sets of elements: RDF itself and RDF Schema (RDFS). Because of the potential ambiguity, when the name "RDF element" is used, it means the first set; otherwise it means RDF as a framework (like in "RDF statement"). The main RDF elements are:

[1] RDF/RDFS/OWL specifications are directly obtained from three standard XML files which have been recommended by W3C to be used as namespace of RDF documents. They are located on these addresses of w3.org website: http://www.w3.org/1999/02/22-rdf-syntax-ns, http://www.w3.org/2000/01/rdf-schema and http://www.w3.org/2002/07/owl

a) *Type*s (used for instantiating a class)

b) *Properties* (used for instantiating a property)

c) *Reification*s (how to write statements about statements)

d) *Containers* (how to build statement about multiple resources).

A general RDF statement has the following look in XML:

```
<rdf:Description rdf:about="thisSubject">
        <thisPredicate>thisObject</thisPredicate>
</rdf:Description>
```

It is also noticeable that XML is only an option for describing RDF Model. There are other alternative syntaxes, like n-Triples [154] and Notation-3 (N3) [24]. The idea of explaining and storing statements as triples is the common approach in all of those languages.

RDF Schema (RDFS) provides some relations and logics to describe concepts, which RDF can use as predefined structures. It is important to notice that unlike XML Schema, RDFS is not used for validating an RDF listing, but it is used for adding more functionality to RDF as well as providing a namespace for that. An XML listing that contains information modelled in RDF has two namespaces: RDF elements and RDFS. RDF elements are used for basic concepts (like types and properties) and RDFS for extended concepts (mostly in an object-oriented framework, as follows).

A summary of RDFS names and meanings are:

a) *Classes* (some resources can be instantiated or be used for inheritance)

b) *Resource*s (a class of everything).

c) Special properties like *SubClassOf* and *SubPropertyOf* to build hierarchical tree of classes and properties

d) Another properties for restriction and validation of RDF statements, like *domain* and *range* of properties

e) Some other descriptive property about resources like *comment* and *seeAlso* (how to describe resources in a free-text formats)

f) Special classes like *Literal*, *Datatype* and *Container* to be used in RDF

OWL is a language about explaining logical relations between resources introduced in RDF elements and RDFS. OWL can be used to validate or logically restrict RDF statements. It can be considered as an extension to RDFS in a higher logical layer. OWL primitives are:

a) Equivalence or difference of resources, using properties like *equivalentClass*, *equivalentProperty*, *sameAs*, *disjointWith*, *differentFrom.*

b) Boolean class combinations, using properties like *unionOf*, *intersectionOf* and *complementOf.*

c) Logical properties of properties, like *TransitiveProperty*, *SymetricProperty*, *FunctionalProperty.*

d) Property inversion using *inverseOf.*

e) More restrictive mechanisms using *onProperty*, *hasValue*, *allValuesFrom…* and cardinality using properties like *minCardinality*, *maxCardinality.*

The above summary of RDF and OWL will be used in section 5.7 to compare the TRM with the RDF, after a TRM definition is provided in the next chapter.

2.6.2 The Main Challenges

One of the early promises made for the Semantic Web is building a global distributed knowledge base [115]. However, there are some pragmatic difficulties for applying the Semantic Web for such a global scope, partially because of dynamic characteristics of the global knowledge and the strictness of the Semantic Web in dealing with human conceptual models [115]. Two main challenging issues are the ability of the Semantic Web to be used with all possible real-life knowledge requirements, and its ability to do that at a global scale [14, 112, 115]. Also because a single RDF triple is about relating two resources by a property, there are many unanswered questions about how to use RDF to express n-ary relations [181].

From the system engineering point of view, two core challenges on the Semantic Web are: 1) Re-engineering the task of semantic enrichment for building the web of meta-data: How this can be done in a high-speed and low-cost manner? 2) Maintaining and adopting such a web, especially considering the dynamic nature of the knowledge: Which knowledge-acquisition methods and machine-learning techniques can be employed? And 3) perhaps the hardest problem to solve is the "ontology mapping problem", when the Semantic Web deals with a multitude of ontologies [14].

Also there are not fixed answers to the questions like: How a human-readable fact must be written as machine-readable? Who must do that? Is that the author of the webpage or some tools? If it is a tool, how trustable it can be? Are all human-readable information is convertible to machine-readable? While humans are flexible in rules and reasoning, how can machines behave so? How to deal with fuzzy rules? [14]

The above issues were about the Semantic Web as a whole, and some more specific challenges about RDF will be studied in section 5.7.1.

2.6.3 The Semantic Web: A Ternary Approach

A similar ternary approach to the Semantic Web clarifies that RDF, as the basic data model of the Semantic Web, uses three URIs to build a relation and the upper layers like OWL use the built ternary relations to accomplish higher degrees of information modelling. By this look, RDF and consequently the Semantic Web have a great potential to be covered by a more general ternary information model. This will be discussed more in section 5.7.

2.7 Workflow Definition Models

The subject of this section is primarily a different field of systems than the previous sections, but with a deeper look, it also possesses another form of node-link structure which makes it a related work to the subject of this thesis. The main related works on workflow modelling will be reviewed in chapter 6.

Workflow systems technology is a growing branch of IT systems that attracts extensive research in recent years. Many of the researches are about unifying the standards, modelling and strengthening the theoretical backgrounds. The position of workflow technology now is similar to the position of the database management systems in early 70's when different people were developing different management systems with different standards and no unified theory could support those works [3].

Processes and workflows have been modelled in several ways and using several notations. The theoretical studies on various modelling types of processes can help building better process management systems, which can consequently help to automate the processes more efficiently, especially in business/industrial processes automation, office automation and e-commerce systems. Some of the benefits of using such automated systems are improvement in speed, quality, reliability and flexibility [52].

2.7.1 Basic Workflow Concepts

Workflow is the sequence of actions or steps used in a process which is usually run by more than one involved parties and uses many different resources [104]. Each multiple-task operation for doing a single goal must have a workflow. Workflows are usually derived from set of operation rules. In complex workflows, some process engineers usually convert these rules or policies to processes, and then a workflow system can handle this process by using computers. Computerizing workflows doesn't mean leaving computers to do the workflow tasks (even if this is possible), but using computer systems to know who must do what, and when it must be done.

There are many advantages and benefits for using automated workflows for business processes, such as improvement in transparency and efficiency, better process control, management, customer service and responsibility, more flexibility to process changes, and establishing paperless and rule-based office environments [42, 52, 68]. Examples are applications enterprise office automation [86], e-commerce [71], e-learning [60] and in general service industry like finance, insurance, etc. [156, 164]. In the field of knowledge systems, workflows help building "rule-based" knowledge management, and are highly combined with concepts of knowledge, especially in representation and solution processes [82, 124].

A Workflow management system (WFMS) is computer support for the design and execution of processes [86], dealing with both defining and executing workflows [107]. It can completely define, manage and execute workflows whose functions are driven by a computer representation of the workflow logic, or a systematic tool for defining and controlling a workflow. The Workflow Management Coalition (WFMC) [195] is the leading body in the workflow community and has a standard workflow management reference model as shown in Figure 2-7. Flexible workflow models are those with clear boundaries between workflow definition and other parts of the model (Interface 1 in the Figure 2-7). Other parts can be considered as some engines to be driven by the workflow definition.

Workflow management systems have some characteristics in common with systems classified as 'knowledge management systems'. Knowledge management ideas can be also added into workflow management for better work within a knowledge organization [83].

WFMS can actively coordinate work processes, manage any condition that can be expressed logically, manage both expected and unexpected conditions and be run on one or more workflow engines. It should have a high level of Interaction with participants, and where required, should invoke the use of IT tools and applications.

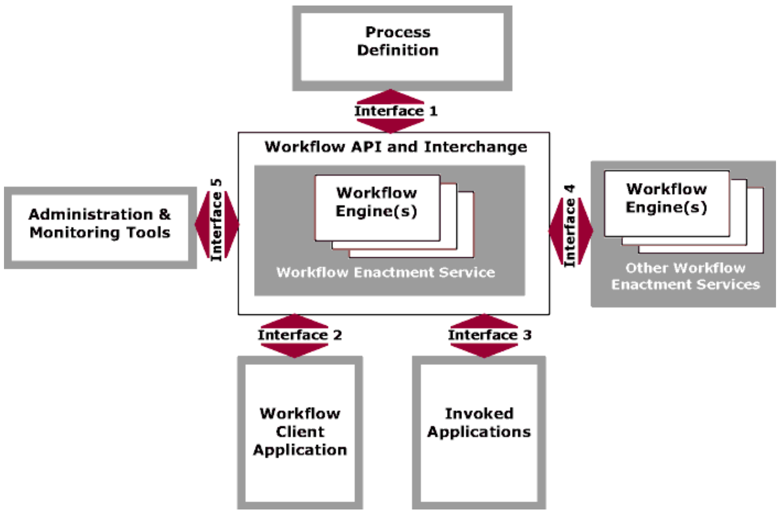

Process Definition
Interface 1
Workflow API and Interchange
Administration & Monitoring Tools
Interface 5
Workflow Engine(s)
Workflow Enactment Service
Interface 4
Workflow Engine(s)
Other Workflow Enactment Services
Interface 2
Interface 3
Workflow Client Application
Invoked Applications

Figure 2-7 : WFMC reference model for workflow management [195]

2.7.2 *Workflow Models: A Ternary Approach*

The rationale behind a ternary approach to workflow modelling is that a workflow has a ternary node-link structure, both in defining and in running modes. The atoms of information in a workflow are some static states that the workflow cases can accommodated (nodes or boxes) in addition to some links between nodes that cases use to move between nodes (relations or arcs). The arcs are carrying the meaning of a case move, so they must be described (by labels or more comprehensively by other nodes). For a single case movement in a workflow, one must be aware of three elements: “from”, ”how” and “to”. These three elements make the studied models to be covered under a general ternary approach. More details of this approach will be explained in chapter 6.

2.8 Ternary Approach to Other Related Fields

There is a general concept of having three basic elements for knowledge atoms in some other fields of information technology. Even in non-IT fields like in linguistics, this concept is evidenced by T-expressions [84]. T-expressions has been introduced as <subject, relation, object> triples and all expressions of a knowledge domain in this theory must be modelled so. A “tense” is stored in knowledgebase as T-expression and other facts of a tense are being stored as its “history”. The T-expression representation is recursive and also T-expressions can be object or subject of another T-expression through some recursive mechanisms [84].

Also the Directed Graphs [72], particularly when used as a knowledge management methods (like in [124]), are expressed as *ternary relations* (like in [33]) when an *edge* in a directed graph is a triple of *(source, label, destination)* and a *leaf* is a couple of *(node, value)*. After defining the TRM in the next chapter, it will be clear that a directed graph can be rewrite in the TRM. The difference between the TRM and this schema is that label and nodes are of different classes of objects while the TRM treats them in a same way. It is then noticeable that the term of “ternary relation” (used as it is for the directed graphs [33]) is more applicable to the TRM than the directed graph. This is because unlike the directed graphs, three same things are related together in the TRM.

2.9 Using the Commonality for Interconnection

The studied facts about finding a possible common ternary approach to the related works may show new or hidden aspects of interconnectivity between those areas. For example, a direct mapping from FOHM (and not any arbitrary open hypertext model) to RDF is possible [87] because both are three-element metadata languages.

In this section, some of such interconnectivities will be shown as a number of case studies.

2.9.1 The Interconnections of Workflow, Knowledge and Hypertext

The case of hypertext-based workflow management systems is an example of knowledge management with hypertext [82] and can be studied as a knowledge-oriented hypertext. Workflow management systems (WFMS) are close topics to both knowledge systems and hypertext. The workflow be applied to manage a corporate or individual knowledge, problem solution process or business process [82]. This will be more clear by noticing that: 1) Workflows have a non-linear nature in task processing and in performing processes; and 2) Knowledge systems, together with business processes, are the main areas of workflow applications [149]. The triangle of supportive relations between these three areas is illustrated in Figure 2-8

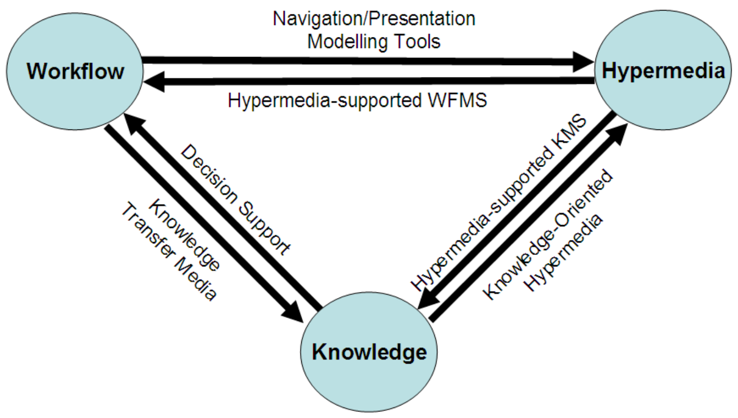

Navigation/Presentation Modelling Tools
Workflow
Hypermedia
Hypermedia-supported WFMS
Decision Support
Knowledge Transfer Media
Hypermedia-supported KMS
Knowledge-Oriented Hypermedia
Knowledge

Figure 2-8: Relations between workflow, hypertext and knowledge systems

2.9.2 *Workflow Interactions with Knowledge Systems*

Workflow Management Coalition [195] defines workflow management as: "Workflow management consists of the automation of business procedures or workflows during which documents, information or tasks are passed from one participant to another in a way that is governed by rules or procedures". Also Patrash's [147] definition on knowledge management is: "Getting the right knowledge to the right people at the right time so they can make the best decision". By mixing these two definitions, workflow management is shown to be able to act as a tool for knowledge management. This has been shown in detail by Garnemark in [82], when he describes how integrating knowledge management techniques with workflow systems can support knowledge collection, storing and sharing. As an example, the knowledge of recognizing a chemical solution has a procedural nature which can be collected, transferred or presented by workflow systems. Noticing that workflows are directed graphs, Collier in [59] shows how a directed graph can represent an specified knowledge.

2.9.3 Workflow Interactions with Hypertext Systems

When studying some interactions of workflows with hypertext systems, two groups of approaches are observable: workflows to help hypertext modelling and hypertext to realize workflow system. These will be studied in the following two sub-sections.

2.9.3.1 Workflows to Support Hypertext

There are some related works in applying workflow concepts in designing and modelling of hypertext systems. The term of "Workflow-driven Hypertext" has been introduced as "the hypertext interfaces that permit the execution of activities and embody constraints that drive the navigation of users" [111]. Mamaani and Abdul Kareem in [114] show that the workflow nature of hypertext when being presented and navigated can be seen as a process, illustrated by some flow diagrams and modelled by workflows and Petri-nets. This idea is a motivation point for some other researchers to bridge between these two domains: Stotts and Furuta in [81, 169, 170] take it to build Trellis model of hypertext based on Petri-nets. Vivekanandan and De Roure in [180] show that open hypertext systems as a set can be modelled with workflow principles for providing better services. However, the resulted workflow is more automatic and less human driven. Collier in [59] shows how directed graphs can be a navigational structure of hypertext systems in their provided system called *Thoth-II*. Brambilla in [36] has integrates the BPMN graphical notations of workflow with WebML notations in order to apply the workflow technology to the conceptual design of an organization's website.

As a recent hypertext modelling approach, "Process-oriented Model of Hypertext" [35] has a process-centric approach to the hypertext conceptual modelling, instead of the data-centric approaches (like WebML, section 2.4.2). As a result, the process-oriented hypertext incorporates some elements from the workflow technology domain (like "users", "groups", "cases" and "activities") into the hypertext modelling. In the process-oriented model, the conceptual design of a hypertext application is divided into hypertext design, data design and process design. Then in the process design, the orientation and configuration of hypertext nodes will help to realize the workflow patterns which are controlling the whole hypertext system.

2.9.3.2 Hypertext to Supports Workflows

There is another group of the related works in employing hypertext environments to support workflow systems. Ashman in [20] suggests that because links can order the processing units of a large process, they can be used to generate workflow models. She has also counted the recording of corporate knowledge as one of the usages of hypertext systems, which is recording the navigation steps of an expert while solving a problem and saving this navigation history as a workflow for future use in order to fill the gap between experts and invoices. In WIS [172], which is a hypertext system and employs Metalevel Links (section 2.4.4) in its link structure, the navigation structure is intentionally designed to support practical workflows. The provided example in that work shows how such a design can support a workflow system to define a recruitment process.

In addition to the definition of the "Process-oriented Hypertext Model" in the previous sub-section, the term "Process-oriented hypertext" has been also used to refer to a class of hypertext deployed to implement workflow management systems in [133]. In this view, hypertext is used as a design platform that can be used to develop information systems, particularly for WFMSs.

2.9.4 The Interconnection of ZigZag and Hypertext

Ted Nelson says: "Zzstructure is not hypertext, while it is composed of nodes and links (like the common hypertext forms), by itself it would make very bad hypertext" [129]. For some others, ZigZag is a paradigm of hypertext [122]. One obvious point is that they are both based on the node-link structure. That is why it is generally accepted that ZigZag is a *hyperstructure*, in which data structures are utilised to model both organizational and presentational aspects of hypertext nodes and links [117]. Also navigational behaviour is obviously involved in both organizational and presentational aspects of hypertext systems. Thus, ZigZag can be an underlying structure for several aspects of a knowledge-oriented hypertext, including the navigational model. This means that in such a hypertext system, nodes are cells of a zzstructure, links are links of that zzstructure, and finally associations of links are dimensions of that zzstructure. It is then clear that the BRM cannot cover the third element and as it will be shown, zzstructure linkbase can not have a binary implementation.

2.9.5 The Interconnection of Databases and ZigZag

As will be explained in section 3.3, it is possible to design a ZigZag information system on top of the database layer. Appendix A contains the details of a ZigZag data navigation system designed using the relational database as its data layer.

2.9.6 The Interconnection of Databases and Workflows

Chapter 7 includes the details of development of a workflow management system on top of a databases layer.

2.9.7 The Interconnection of Databases and Hypertext

DHE [29, 30] (described in section 2.4.5) was an example of building hypertext links over a relational database. Moreover, databases and hypertext system can have other forms of supporting to each other, which DHE is only an example of those approaches. Hypertext can also be used as a user interface to retrieve information of databases [28, 80, 140] and in terms of usability, this kind of user interface for a database system is more efficient than the traditional tabular approaches [120].

2.9.8 The Interconnection of Databases, XML and Directed Graphs

There are methods of building relational databases from both directed graphs and XML [33, 77, 78] by several mapping methods. Also databases is counted as one of alternative ways of XML storage strategies in [175]. A review on those works simply shows that the directed graphs are usually used as an intermediate stage to map between XML and databases, and also the ability of converting XML to directed graphs and database is based on the existence of a common ternary foundation.

2.9.9 The Interconnection of ZigZag and Directed Graphs

Zzstructure can be defined in several ways, including a definition based on directed graph in [116]. In that view, zzstructure is a "directed multi-graph" with some extra restrictions. This is concluded by means of a ternary approach; however, a pure ternary definition of zzstructure has been introduced in section 2.3.3.

2.10 Summary

In this chapter, through reviewing the related works, the direction was to find a "ternary" common foundation in some different areas: Knowledge management, hypertext, the Semantic Web, ZigZag and Workflow management. What is meant by the "ternary foundation" has not been formally defined yet and it is supposed that the reader in this stage has enough background and motivation to read chapter 3 and to know the basic definitions of the TRM, having an implicit view about what are the expected properties of the TRM.

So, we now return to our original question – are we talking about the same structure? Although this chapter could provide a rough idea about the targeted unified model, it yet cannot accurately define what that unified model is. The fundamental information model of all of these paradigms has been shown to be a "ternary node-link structure" [150] but this needs to be justified through this thesis after introducing TRM in the next chapter, particularly in sections 3.3, 4.1, 5.7, 6.4.

Chapter 3 -

THE TRM: A GENERAL INTRODUCTION

In chapter 2 an implicit view of a common information model has been drawn. This model is based on three elements that can be used in a variety of node-relation structure. It is supposed that chapter 2 could justify the basic need for "association" as a main element that can be modelled explicitly in nodes of the structure neither in the relations nor implicitly anywhere else. Thus the drawn image of the proposed model must contain three elements of "source", "association" and "destination" as the relations atoms. This model is called "Ternary Relations Model" or "TRM" to express that it is based on relations between three nodes, or some triples which are related together. This chapter is to introduce the TRM in an abstract context and to show how it can be a generalization of all "ternary approaches" studied in the previous chapter.

3.1 Abstract Definition of the TRM

The definition of the TRM is proposed in two stages: Static and Dynamic. The Static TRM is applicable when the relating nodes are fixed and independent of each other and/or external parameters, and the dynamic definition extends the concept of the Static TRM to the areas where nodes can be functionally dependent of each other.

3.1.1 Static TRM Definition

The basic concepts are:

1- "Node" is a name for every individual piece of data/information.

2- A "Relation" is an ordered triple of any three nodes called "source", "association" and "destination" respectively.

3- Each relation is itself a node.

4- "Relations" in the TRM are bi-directional, which means that a single relation can express two meanings when being read or interpreted from two different directions.

The TRM information space is as simple as a set of nodes. There is no hierarchy structure between nodes in the TRM. The fact that "relations are themselves nodes", doesn't imply any hierarchy because it can be resolved by cross-referencing.

The TRM graphical notation consists of circles representing nodes and arrows representing relations. Arrows originate from the source node, passing through the association node and terminate to the destination node. This has been illustrated in Figure 3-1, and a sample TRM-modelled information space has been shown in Figure 3-2.

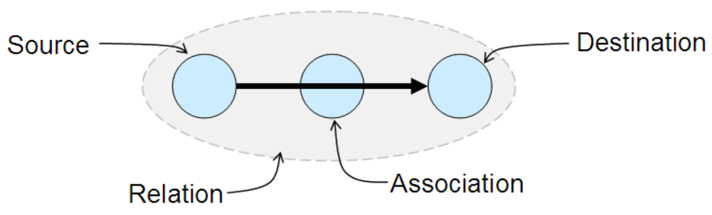


Figure 3-1: Graphical notation for the abstract TRM definition

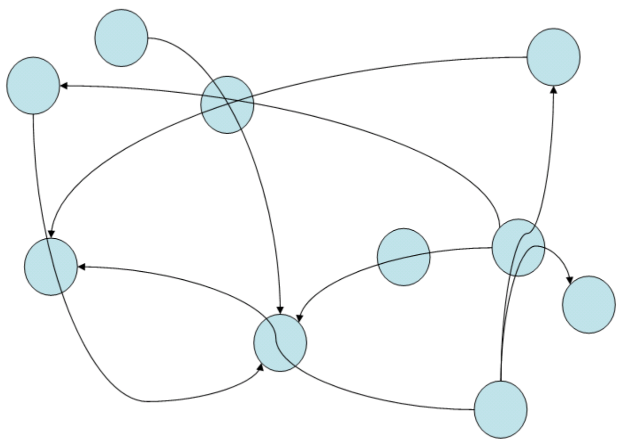


Figure 3-2: A sample TRM-modelled information space

It is very important to clarify a point about the TRM notation. "This notation is not supposed to be used instead of any other notation or language or to be a competitor to them". As will be seen later, for each individual subject of work, like workflows or XML, some equivalent TRM notation exist, but it doesn't mean that the TRM notation is recommended. The target is to say that "it is possible" for the TRM to express them. This is because the TRM notation is going to be cumbersome for large amount of information. However, it may or may not be a rival notation depending on the subject.

3.1.1.1 Formulation

The fact that "each relation is itself a node" is a recursive phrase in formulating the TRM and may look to cause problems in making a closed TRM formulation. Fortunately, this recursion doesn't imply any infinite loop because there are some basic nodes, i.e. nodes that don't contain any relation, otherwise it would be impossible to build relations over other relations. This has also the advantage of self-description, in which one can say "everything in the TRM is node" to make the TRM definition very simple.

Let us first assume that relations are different entities from nodes. The TRM is defined as:

A couple of (N,R) where
N is the set of all nodes and
$R \subset N^3$ *i.e.* $R=\{\textit{some } (x,y,z) \mid x,y,z \in N\}$

Now for imposing the fact that every relation is itself a node, it is not possible to simple say $R \subset N$ because N doesn't include triples by definition. So let us define N_0 as the set of basic nodes, i.e. nodes which doesn't express relations. Then N is the set of all nodes which is the union of N_0 and R. In this case, R is N^3 and not N_0^3 because relations can be built over other relations as well as basic nodes. Now the Static-TRM is defined as:

A set of all nodes: $N=N_0 \cup R$ *where* N_0 *is a set of basic nodes; and* $R \subset N^3$ or,

$$\boxed{N=N_0 \cup \{\textit{some } (x,y,z) \mid x,y,z \in N\}}$$

Equation 3-1: Static-TRM formulation

It is noticeable that *N* is defined using *N*, which expresses the recursive definition. This formulation shows a *node production machine*: Having a set of N_0 is enough to grow the information structure in the TRM, all one needs to do is to build more triples and add them to the existing nodes.

3.1.1.2 Internal Architecture

In the TRM, each node *can* have the following data members:

1- *Id* and/or *URI*

2- *desc* (description)

3- *da* (direct association) and *ra* (reverse association)

4- *src* (source), *asc* (association) and *dst* (destination)

Id is the unique internal identifier of the node which can be used to reference to any node. The validity of *Id* can be defined in different scopes, which can be universal or local, depending on the application. *URI* defines the web standard identifier of the node. *desc* is a text containing the name, description or value of the node, independent of possible roles of the node in any relation. *da* and *ra* are two texts describing two faces of this node when it participates in a relation in normal or reverse directions. *src*, *asc* and *dst* are references to *Id*s of three other nodes and are used when this node is a relation (or a statement) about the other nodes.

Id/URI is the only necessary data members and the rest of members are optional. This has been intentionally defined in order to allow the node structure to handle both single node definitions (when at least one of *desc*, *da* or *ra* is needed) and relation definitions (when *src*, *asc* and *dst* are needed).

3.1.2 Dynamic Definition

The TRM in general supports the functional links, which means that each of the three elements of a relation can be a dynamic function of another. It also means that they are not only changeable by the authors, but also they can be changed dynamically on the reader's side. Thus in the most general case, each of the three elements in a ternary relation can be a function of two others and the environmental attributes on the reader's side (like user's specifications, location, time, etc.).

This functionality will allow us to cover some areas which cannot be covered by the Static-TRM, like the functional links of the BRM, as mentioned in section 2.5. Also by mixing this functionality to some of the related works studied in chapter 2, new horizons may be opened to extend those information models and build new models. As an example, ZigZag if mixed with the functional links. It is noticeable that the functional links of the TRM act in a totally different level than the TRM links level. It means that the links are still ternary and the functional links are not intended to support extra link dimensions if they are needed. Instead, the functional links generalize the way that three nodes can be linked. In the Static TRM, a link is about three fixed nodes and in the Dynamic TRM, it is about three variable ones. As will be seen later, the dynamicity is neither about the number of nodes to be linked, nor about the content of each node, but about selecting the participating nodes. The content of a node is not necessarily fixed in the Dynamic TRM (e.g. a functional link is itself a node with dynamic content) but a single functional link does not act on the level of changing the content of the nodes.

3.1.2.1 Formulation

In order to formulate the Dynamic-TRM, three functions with some attributes must be defined. Also it is necessary to have an abstracted attribute to show all of the environmental parameters. Making this attribute is completely dependant on the nature of information and may be different from case to case. Thus in the following formulation, it is assumed to have an abstracted and single parameter, named *t*, which includes all of the necessary environmental parameters. A set of all possible *t*'s is named T.

The Dynamic-TRM is defined as follows: the Static-TRM (section 3.1.2.1) plus:

N is the set of all nodes

$R \subset N^3 \quad$ *i.e.* $R=\{\text{some } (x,y,z) \mid x,y,z \in N\}$

$T=\{\text{some } t \mid t \text{ is an environmental parameter}\}$

$x=f(y,z,t)\,,\ y=g(x,z,t)\,,\ z=h(x,y,t) \quad ;\ \text{and} \quad f,g,h\colon (N^2 \times T) \rightarrow N$

Equation 3-2: The Dynamic-TRM formulation

In terms of notation, it is difficult to draw graphs for Dynamic-TRM because the result would be a dynamic graph that changes by having different environmental parameters.

However, dotted line has been used in a simple example shown in section 7.1 to denote a time-dependence link. This solution may not be applicable in a more complex example.

Relating three nodes to each other while the relation uses functions needs more clarification. A possible misunderstanding is to suppose that nodes are themselves dynamic in content, and that the Dynamic-TRM relates these varying nodes to each other. Although it is possible for the contents of nodes to be changed at any time (like any other information model), the Dynamic-TRM has nothing to do with the changes in the content of nodes. Being dynamic here is about relating nodes, not about contents of the related nodes. If a single defined relation relates three fixed nodes of x,y,z_1, it is possible that under other circumstances the same relation relates x,y,z_2. So the TRM relations include references to some three nodes, while the mechanism of referencing is direct in the Static-TRM and indirect (functional) in the Dynamic-TRM4. The practical solution to this is to define the functions like *f*, *g* and *h* (of

Equation 3-2) as nodes and use them as source, association or destination of the TRM relations. One then can have a static snapshot of the Dynamic-TRM by knowing the result of the dynamic functions.

3.2 Examples of the TRM

3.2.1 The Static-TRM

As an example of the Static-TRM, the bibliographic example of section 2.1.2 is recalled. However, the information in that example doesn't need all of the Static-TRM features (like bi-directionality). Figure 3-3shows the equivalent TRM graph.

4 It is obviously possible that the content of a link is changed (an example is a node representing a dynamic TRM link – as a relation is itself a node) but that node still has a fixed identification that make it ready to participate in any another static or dynamic link. The latter dynamic link again has nothing to do with the changes inside the participating nodes, even if they are themselves changing.

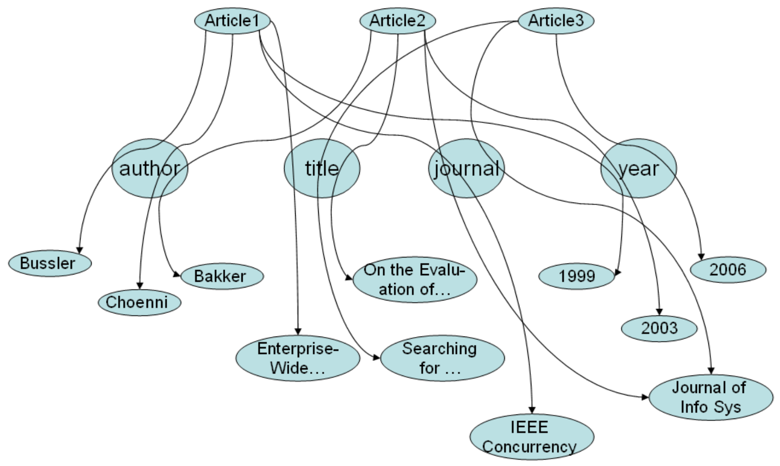


Figure 3-3: The TRM graph representation of the sample database

3.2.2 The Dynamic-TRM

Ternary-Links in a personalized hypertext (section 5.6) is an example of the Dynamic-TRM, and the environmental parameters are the identified user specifications and possibly time. The system may show an anchor in the hypertext as the source of a link while this could not be a link with other users, then by clicking on that anchor a menu of choices may appear that shows the available links while each item may vary for different users. By selecting one of them, another environmental parameter is involved which is the "user's choice". Finally another function may calculate the desired destination with all of the available parameters during the process and will take the user to that point.

DHE [29, 30] (described in section 2.4.5) is another special case of the Dynamic-TRM when the added computed links are in the forms of functional triples. The rules of finding links in DHE (called RNA: Relationship Navigation Analysis) are based on the non-normalized schema of a relational database. The set of links in the resulted hypertext is $R=\{(x,y,z) \mid y=f(x), z=g(x,y)\}$ when x is the content of a field, z is the available semantic relationships originating from x, and z is the endpoint. Finally, *f()* and *g()* are functions generated by RNA.

3.3 A Layered Approach

After defining the TRM in this chapter, it is necessary to find out how TRM fits in with other related works studied in the previous chapter. What has been used till now was the word "covering" to show that the TRM can be a common foundation for different information models. This word must be clarified to show the level of such coverage. For example, the way that the TRM covers the BRM is far different from the way that it covers Zzstructure.

To achieve the explicit relation, a layered definition of an Information System has been proposed in Figure 3-4.

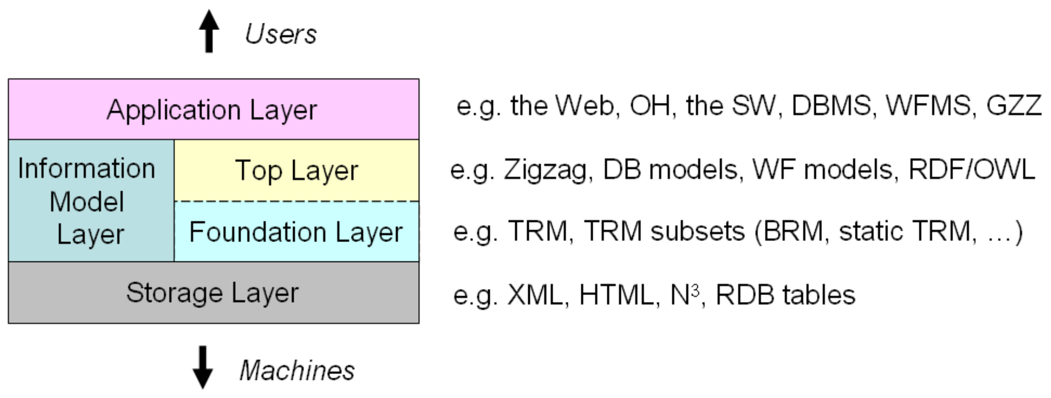

Users
Application Layer
Information Model Layer
Top Layer
Foundation Layer
Storage Layer
Machines
e.g. the Web, OH, the SW, DBMS, WFMS, GZZ
e.g. Zigzag, DB models, WF models, RDF/OWL
e.g. TRM, TRM subsets (BRM, static TRM, …)
e.g. XML, HTML, N3, RDB tables

Figure 3-4: Information System Layers used in this research

In that configuration, the upper layers are the closer ones to representing the information to the user and the lower layers are the closer ones to the machine physical level. Each layer provides enough tools or functionality to represent the information provided by its underlying layers. The layers are introduced as:

1. *Storage Layer*: Contains mechanics, vocabularies or syntaxes about how to store data in files.

2. *Information Model Layer*: Deals with the method of structuring the information in a space of information, from raw data to user-level information.

   2.1. *Model Foundation Layer:* Deals with how to build the structural units of information using raw data.

   2.2. *Model Top Layer*: Deals with how the structural units can be managed to build the user-level information.

3. *Application Layer*: Provides functionalities or tools for users by managing the user-level information.

Now the TRM is positioned in the Model Foundation layer, which means that the TRM deals with building the basic information units. The TRM subsets -with less degree of freedom than the defined TRM- like the BRM or the Static-TRM are also basing some related top-layer models and they are in the same layer as the TRM.

ZigZag, relational and semi-structured databases, workflows models, and RDF/OWL are categorized in Model Top Layer because they know how to manage the information units in an information space. Members of application layer manage the user-level information like the World Wide Web, open hypertext systems or workflow management systems. Going downward, the Storage Layer provides the storage rules, syntaxes or vocabularies, like XML as a textual language to store information in semi-structured database. Tables of a relational database (including the mechanics of how the information is arranged in tables) are also categorized under this layer.

It will be shown in section 4.1 how XML can be converted to the TRM graph. Now the question is how something from a lower layer is to be converted to something from an upper layer? The answer needs a deeper look on what has been converted, and as will be shown will result in another interesting outcome. What is converted to the TRM graph is not the vocabulary or syntax of an XML listing, but it is some semi-structured information (or a hierarchy of information). That information could be written in a few possible languages, including XML. Since XML is the most common way of expressing such kind of information, it seems that XML has been converted. The TRM is located in an abstracted layer over XML and RDB tables, so it shows that the TRM may be expressed in XML or tables. In fact it will be shown in the next chapter that the Static-TRM can be expressed both in XML and RDB tables without any contradiction. "TRM-XML" will be introduced as a language of expressing the TRM-based information in XML. This naming is not because the TRM-XML is not XML or not because it has not the XML syntax, but because of its special vocabulary. An instant and confusing result is that XML can be rewritten in the TRM-XML (which is still in XML); something that looks like a recursion, but in fact is changing the information modelling method. The other outcomes of this fact are left here to be studied after introducing the TRM-XML in the next chapter.

This layered orientation can also be evidenced by DIKW pyramid explained in section 1.1.1. and illustrated in Figure 1-2: The storage layer represents "data", the information model layer represents information, and the application layer represent knowledge. In the information model layer, the foundation layer (here TRM) builds information by relating data, and the top layer (like RDF) makes patterns of information to be represented as knowledge in the application layer.

The order of layers is not about the richness of information but about their position in the user/machine interactions. Particularly, the role of the foundation layer can be expressed as "making information from data" and the top layer as "making knowledge from information" by reference to the data-information-knowledge hierarchy discussed in chapter 1. The fact that the TRM has more features than (for example) ZigZag doesn't mean to swap their level in that figure. Instead, the fact that ZigZag uses a (subset of) TRM features in making its building blocks leads to put it on top of the TRM.

It is noticeable that the TRM itself does not necessarily or directly involve in user's side or in machine's side. In other words it is not a direct visualization tool, nor a machine coding method. The value of TRM is benefiting the user from the values of a ternary approach to links implementation.

3.3.1 Bottom-up Threads

As an outcome of the mentioned layered approach, there are some Bottom-up layer threads to be discovered, in which not all of the layers to be covered necessarily nor all of the members of a layer can serve all members of an upper layer. For example the BRM may not serve ZigZag, or tables may not server the Dynamic-TRM. But there are some possible threads to be counted here:

1- Tables→TRM→DB→RDBMS: Shows a usual relational database management system.

2- Tables→BRM→DB→OH: Shows an open hypertext system with binary linkbase stored in a relational database tables.

3- HTML→BRM→…→Web: This shows the status of the Web with normal binary links.

4- XML→TRM→RDF→SW: Shows the status of layers in the Semantic Web: This clarifies that the TRM and RDF are in two abstracted layers.

5- XML→TRM→…→DBMS: Is it possible to build databases over the pure TRM? A good starting point to chapter 4.

6- XML→TRM→DB→OH: Building open hypertext systems with some Ternary-Linkbases, a motivation to move to chapter 5.

7- Tables→TRM→ZigZag→…: Shows a zzstructure stored in tables, as demonstrated in Appendix A.

8- Tables→TRM→WF→WFMS: Shows a WFMS based on ternary relations which is used tables of a DB as storage. This is what has been implemented and explained in [152]. The same idea can be done based on XML as well.

3.4 Summary

In this chapter the TRM theory has been formed. The TRM is introduced to be a collection of non-hierarchical nodes. The concept of relations (which themselves are nodes in the TRM) are based on triples. Two versions of the TRM called static and dynamic are formulated: The Static-TRM for fixed triples and the Dynamic-TRM for ternary functional links.

A layered structure has been introduced which can precisely locate the TRM among other works and information models. The TRM is shown to be located in the foundation layer of information modelling techniques, while being on the top of logical and physical storage layers. Tracing possible bottom-up threads in those layers helps justifying some implemented works as well as discovering some unimplemented ones. It can also be a good motivation for moving to the next chapters in order to build new information models.

According to these arguments, the TRM is a highly generalized approach to information that may be used to unify many existing information models. In effect, it may be viewed as an *Information Model Construction Kit* for the next chapters.

C h a p t e r 4 -

TRM-DB: A NEW SCHEMALESS DATABASE

There are many situations in IT systems where people need to manage real-world information and desire not to be constrained by “schemas”. While real-world information is free in structure, the traditional desire in computer world was to store information in some rigid structures. These rigid data structures were developed to serve business in the early computer ages, and as such their design is, in many respects, a direct descendent of hundreds of years of bookkeeping [47].

Spreadsheets, Relational and Object-Oriented Database Models are all about table-orientation and/or hierarchy and are based on the dependence of data to some kinds of associated schemas. Despite all of the benefits of these technologies, fitting the real-world data to the associated schemas has been always together with many challenges on how to artificially rearrange data, how to show them in natural ways and more importantly, how to dynamically apply structural changes. Having two separated sides -data and metadata- for a database management system implies keeping a permanent gap between designing and using the database systems. The more dynamic the data is, the more difficulties in managing these two sides are likely to appear.

The basic idea of this chapter is introducing a very general database model based on the TRM, called “TRM-DB”. Every piece of data and the relations between them in the TRM-DB have a single and global type, called ‘node’. As will be described, because there is no associated schema to a specified data set, it will be called “schemaless” here.

4.1 An Overview of the TRM-DB

Recalling from the layered approach proposed in section 3.3, the TRM-DB is located in the Model Top Layer in the group of DB models. It is supposed to be a database on top of the TRM (as the Model Foundation Layer) that can use the full potential of the Static-TRM, not a subset of that. Thus the TRM-DB is some tools to manipulate information in a complete Static-TRM framework. As shown in the layered design of Figure 3-4, the storage layer of the new database system is some known storages like tables or XML. This section proposes implementing the TRM-DB on top of those two storage layers. These two implementations are called "TRM-Table" and "TRM-XML" hereafter and will be introduced in the next two sections.

Before introducing the implementations of the TRM-DB, it is necessary to study how the TRM can be formally the fundamental information model for the studied related works, including RDBs, XML and ZigZag (as claimed in Figure 3-4).

4.1.1 The TRM-DB behind the Relational Databases

The TRM can be extracted from any data modelled in the relational databases. In fact, RDBs have their own method to making ternary relations: "Tables". Two approaches are possible to explicitly express RDB tables in the TRM: In the first approach, tables are viewed as expressing predicates, and the second approach uses the binary decomposition to relate RDB tables to the TRM.

Firstly, a table can be viewed as a single semantic predicate (or association, in TRM term) between a record identifier and a record, and a record itself is a set or ternary relations between a record identifier, a field name and an individual data sit in that field. In other words, tables are semantics that relate tables to records, and the field names are semantics that relate records to data. For example, a table of "articles" with fields such as "title", "author",etc. is a set of ternary links like :

{(article_id1, articles, (article_id1, title, title1)),
 (article_id1, articles, (article_id1, author, author1)),
 (article_id2, articles, (article_id2, title, title2)), … }

Secondly, can be easily proven by noticing that each data modelled in relational database can be decomposed to a set of binary relations [55]. It means that after decomposition, there will be an infinite number of two-column tables that include all of the information necessary for rebuilding the original database. The two columns are usually IDs and textual strings. Thus there will be a number of relations called R_1 to R_n where:

$R_i = \{ (x,y) \mid x \text{ has relation } r_i \text{ to } y \} \; ; 1 \leq i \leq n$

Then R_0 can be defined as:

$R_0 = \{ (i, r_i) \mid 1 \leq i < \leq n \}$

Finally a general set of R can be defined as:

$R = \{ (x,i,y) \mid (x,y) \in R_i \, , 0 \leq i \leq n \}$

The above set is a kind of the Static-TRM formulation, according to Equation 3-1. This conversion has been illustrated in Figure 4-1.

It is noticeable that the binary decomposition may be widely impractical, as it may end up with an uncontrollable number of relations, but has been used here to support the TRM theory.

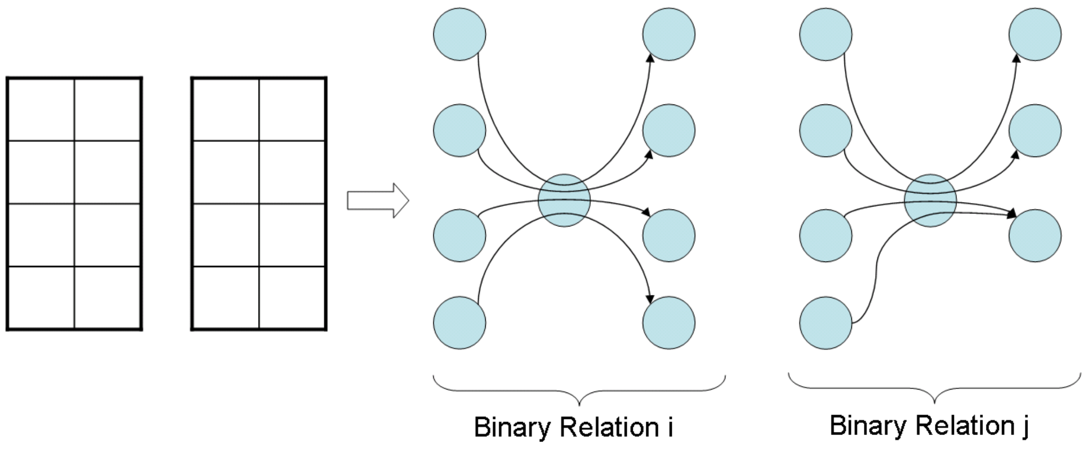


Figure 4-1: A sample conversion of a set of binary relations to the TRM graphs

4.1.2 TRM vs. XML

According to section 2.2, an XML listing can be viewed as a set of ternary relations between elements, attributes and textual values. The main idea is that an XML tag is an association between its super-element and the entire sub-element or the element‘s textual content. Also attribute names are associations between elements and textual values. This shows that an XML listing can be converted to the TRM graph. Also the TRM has some different properties from XML: The TRM is free from any hierarchy; it supports functional linking and bidirectionality.

The idea of converting an XML listing to a TRM equivalent includes making ternary relations between an entire element to its contents (whether sub-elements or attributes). For relating a node to its sub-elements, the required associations are the name of sub-element and the name of attributes. The entire elements (and sub-elements) are themselves nodes that have no explicit equivalent in an XML listing (The name of the element is not a good candidate because the elements can be repeated and the name must not be re-used for each occurrence of an element). Thus some temporary nodes (like n1, n2 in the following example) must be used. Finally the elements in the first level of hierarchy are connected to the whole XML document (named "root") via their element names.

To shows how to convert an XML listing to a TRM graph, these steps must be carried out:

1- A node called the "root" is defined.

2- All tags and sub-tags are representing by some nodes.

3- The listing between each opening and closing tag (an element or a sub-element) is also represented as a node.

4- All attributes are represented as some nodes.

5- All textual values (either for attributes or for elements) are also represented as some nodes.

6- The root node is connected to the nodes representing first-level element via the nodes representing the first-level tags.

7- The first level elements are connected to the nodes representing the second-level element via the nodes representing the second-level tags.

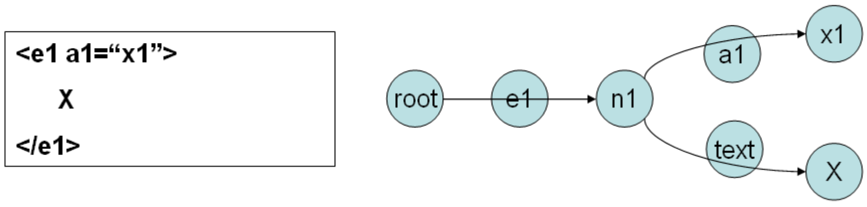
8- Repeat step 7 for all the nested elements.

9- For all attributes, the relevant node is connected to the relevant textual value node via the relevant attribute node.

10- A node called "text" is created, and then all the element nodes are connected to the relevant textual value node, if they have any.

It is also noticeable that during the above process, there may be repeated tags, attributes or textual values, which must not be represented as different nodes, and the maximum reuse must be utilized. In addition, XML supports *ID* and *IDREF* couples to make cross-referencing. It will be very easy to represent that in the TRM graphs by having textual value for IDREF: attributes must not be created as nodes, because they have been already created. Thus the referencing element node must be connected to the referenced element node via the relevant attribute name. In this case, the node "id" representing the ID attribute is a special pre-defined node.

To illustrate the above points, two examples are shown in Figure 4-2 and Figure 4-3. The first figure is to illustrate the main idea in a simple example and the second figure includes how to convert sub-elements and cross-referencing to the TRM graphs.



```
<e1 a1="x1">
    X
</e1>
```

Figure 4-2: A simple example of converting XML to the TRM graph

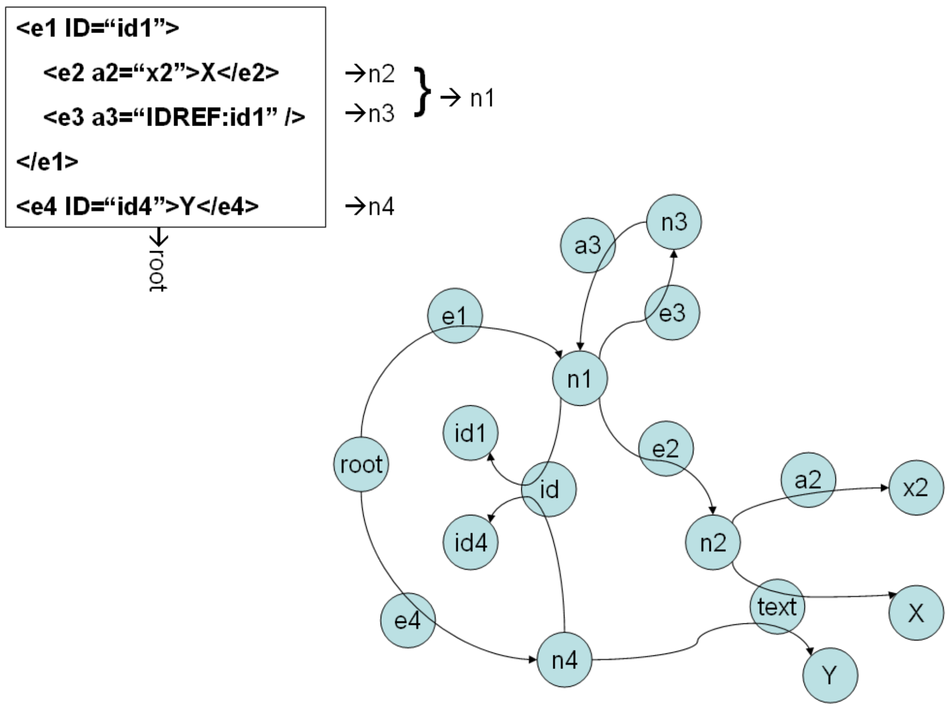


Figure 4-3: An example of converting XML to the TRM graph considering sub-elements and ID referencing

4.1.3 The TRM vs. ZigZag

According to section 2.3.3, a ternary formulation on zzstructure has been defined. That formulation consists of a triple of (*C*, *Z*) where *C* is the set of all zzcells and that $Z \subset C^3$, plus two extra conditions about the uniqueness of right and left connections along a single dimension. Comparing that formulation to Equation 3-1 and
Equation 3-2, it is clearly concluded that zzstructure formulation is a special case of the TRM formulation. The difference -or what the TRM has over zzstructure- are:

1- The TRM supports multiple connections through a single association (zzdim here), i.e. the TRM formulation doesn't imply such extra conditions about the uniqueness of right and left connections along a single dimension. This is also a solution to the problems of one-to-many relationships in ZigZag explained in section 2.3.1.
2- Zzstructure's relations are not themselves nodes (zzcells here). This prohibits zzstructure to be able to built relations over relations.
3- TRM nodes can be repeatedly used in different TRM relations without any need for transclusion; however, the TRM can implement transclusion if needed (e.g. to simulate ZigZag) by connecting through special node of "d.clone".

Also according to the TRM internal structure explained in sections 3.1.1.2 and the TRM dynamic definition in section 3.1.2.1, two other differences between the TRM and Zzstructure are:

4- Zzstructure does not support bi-directional links, i.e. a zzdim has only a single description along its positive direction, and there is no way to realize the explicit meaning of the connection from the destination cell to the source cell. The TRM can fulfil the ambiguity problems of ZigZag explained in section 2.3.1.

5- ZigZag cells are enumerated and it does not support the functional links, thus it can only be a under the category of the Static-TRM.

As a result, a zzstructure graph can be converted to the TRM graph (but not vice-versa). The fact that "two zzcells can be connected along a zzdim" is mapped to the fact that "two TRM nodes can be connected through an association". In order to do the conversion, one needs to define separate nodes for both zzcells and zzdims, and connect them in the same way. For cloned cells, one can either translate them directly to the TRM (by relating via a special node of d.clone), or to redesign the structure in the TRM (by re-using a single node). A sample conversion between two graphs has been illustrated in Figure 4-4.

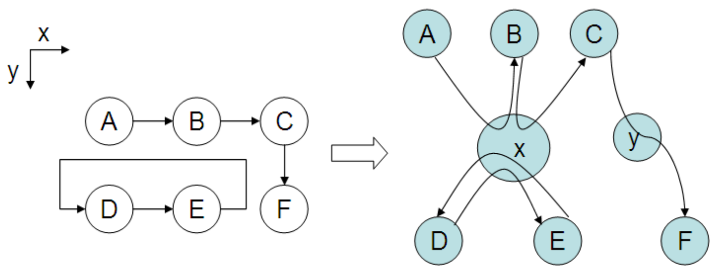
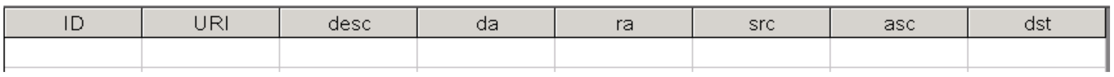


Figure 4-4: A sample conversion of Zzstructure to the TRM graph

4.2 The TRM-Table

The TRM-Table uses a single table to store the entire database (called the "Nodes" table). The singularity of the table is the core of the TRM-Table, as it is enough to manipulate data without any data-related schema, hierarchy or relation between different tables (like in RDBs). It still can be managed in a relational database engine and be queried using languages like SQL, because it is basically nothing more than a table. The distinction is "how" and "what" to store in the table, because it is able to store any data modelled in the Static-TRM.

The table design is simply like Figure 4-5



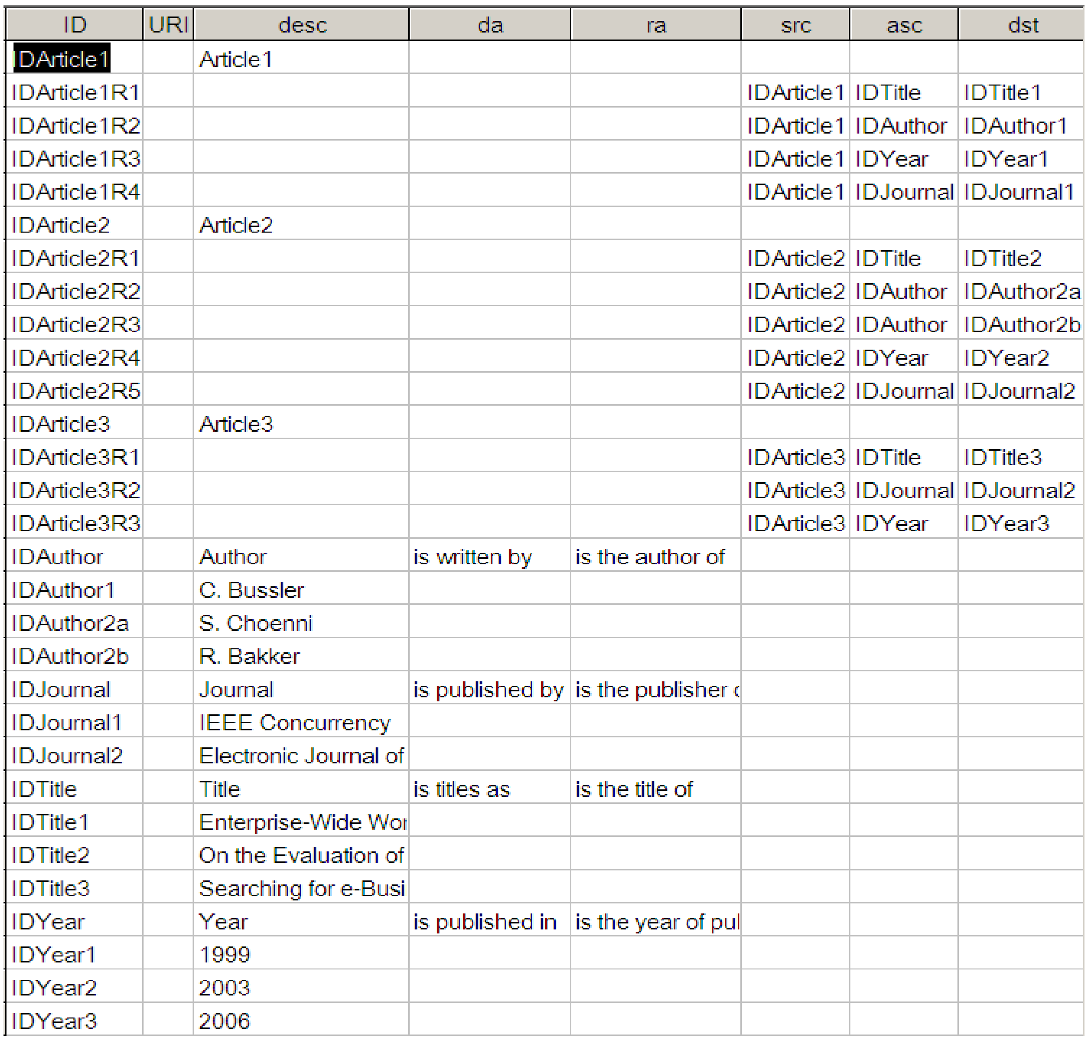
ID	URI	desc	da	ra	src	asc	dst

Figure 4-5: Design of the TRM-Table called Nodes table (all fields are text)

Although the above design is a table, it is still called "schemaless" in this context. This is because the field names are independent of the data and are originated by the information model, not by the information itself. By this view, the tabularity of the design does not imply any rigidity on the handled information. Thus the TRM-Table may be considered as an "Irregular Table".

4.2.1 An Example

Recalling the bibliographic database of section 2.1.2, the equivalent TRM-Table (called Nodes) is shown in Figure 4-6.



ID	URI	desc	da	ra	src	asc	dst
IDArticle1		Article1					
IDArticle1R1					IDArticle1	IDTitle	IDTitle1
IDArticle1R2					IDArticle1	IDAuthor	IDAuthor1
IDArticle1R3					IDArticle1	IDYear	IDYear1
IDArticle1R4					IDArticle1	IDJournal	IDJournal1
IDArticle2		Article2					
IDArticle2R1					IDArticle2	IDTitle	IDTitle2
IDArticle2R2					IDArticle2	IDAuthor	IDAuthor2a
IDArticle2R3					IDArticle2	IDAuthor	IDAuthor2b
IDArticle2R4					IDArticle2	IDYear	IDYear2
IDArticle2R5					IDArticle2	IDJournal	IDJournal2
IDArticle3		Article3					
IDArticle3R1					IDArticle3	IDTitle	IDTitle3
IDArticle3R2					IDArticle3	IDJournal	IDJournal2
IDArticle3R3					IDArticle3	IDYear	IDYear3
IDAuthor		Author	is written by	is the author of			
IDAuthor1		C. Bussler					
IDAuthor2a		S. Choenni					
IDAuthor2b		R. Bakker					
IDJournal		Journal	is published by	is the publisher o			
IDJournal1		IEEE Concurrency					
IDJournal2		Electronic Journal of					
IDTitle		Title	is titles as	is the title of			
IDTitle1		Enterprise-Wide Wor					
IDTitle2		On the Evaluation of					
IDTitle3		Searching for e-Busi					
IDYear		Year	is published in	is the year of pul			
IDYear1		1999					
IDYear2		2003					
IDYear3		2006					

Figure 4-6: the TRM-Table equivalent for the bibliographic example

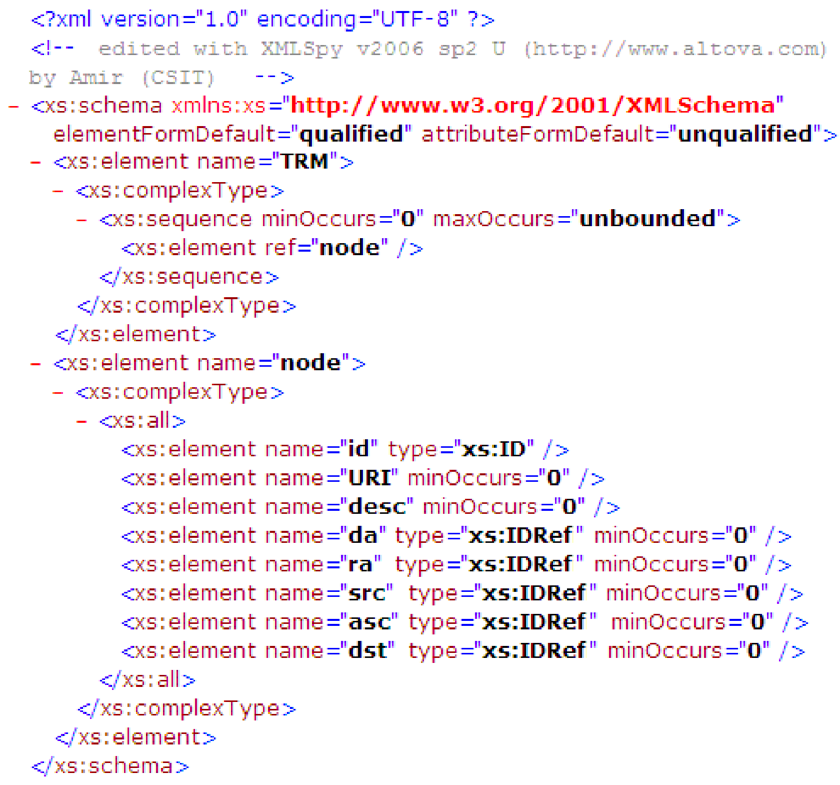
4.3 The TRM-XML

A fundamental rule in the TRM is that every piece of information is a node and there is no hierarchy of nodes. Although XML is designed to manage hierarchical data, it can be adopted here to manage the TRM nodes in a one-level of hierarchy. The TRM-XML is then an XML-based "language" that is used to express information modelled in the Static-TRM. This language uses XML as syntax and the TRM rules as its vocabulary. Recalling from section 3.1.1.2, the internal data structure of the TRM nodes motivates to use XML to assign textual or referenced values to sub-elements of a node element. By this view, the sub-elements of a node element are same as what has been introduced in section 3.1.1.2 as *Id/URI*, *Desc, da*, *ra*, *src*, *asc* and *dst.* As described in section 3.1.1.2, TRM does not require all links to be bi-directional, so *Id* is the only necessary sub-element and other sub-elements have been intentionally selected to be optional.

For its vocabulary it needs a dedicated schema. Like any other XML, the schema can be expressed in another XML file called the TRM-XMLSchema. Similar to what has been mentioned about irregularity of the TRM-Table, the TRM-XML Schema does not imply any rigidity (unlike any other XML Schemas). The TRM-XML Schema is a unified "schema" for the "schemaless database".

4.3.1 The TRM-XML Schema

The basic element is called <*node*> with sub-elements including *Id*, *URI*, *desc, da, ra, src, asc, dst.* Having sub-element is not against the TRM's "no hierarchy" principle, because these are sub-elements of a "node" and are designed to express the internal data structure of a single node. Figure 4-7 shows the list of the TRM-XML Schema.



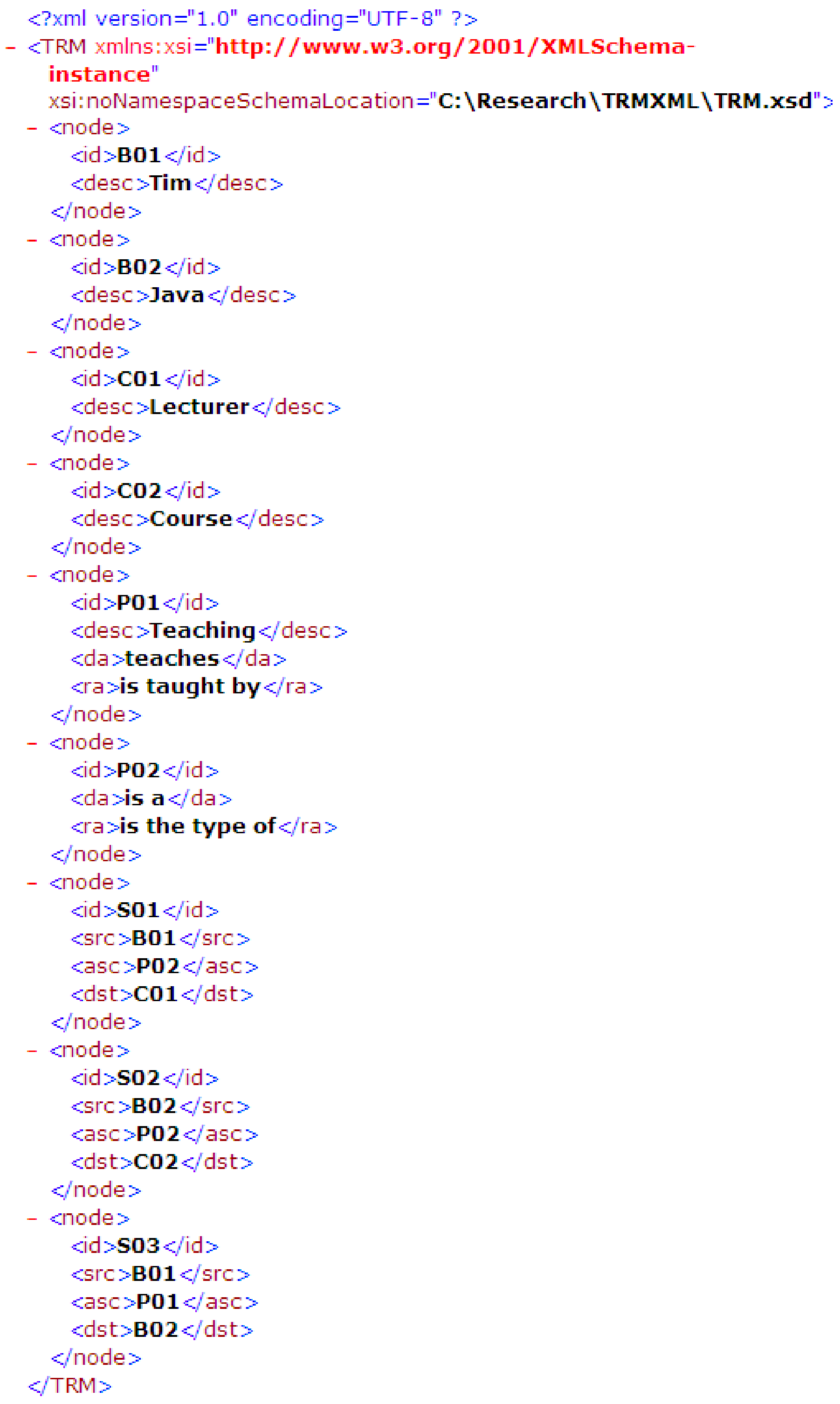
```
  <?xml version="1.0" encoding="UTF-8" ?>
  <!--  edited with XMLSpy v2006 sp2 U (http://www.altova.com)
  by Amir (CSIT)   -->
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <xs:element name="TRM">
    - <xs:complexType>
      - <xs:sequence minOccurs="0" maxOccurs="unbounded">
          <xs:element ref="node" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  - <xs:element name="node">
    - <xs:complexType>
      - <xs:all>
          <xs:element name="id" type="xs:ID" />
          <xs:element name="URI" minOccurs="0" />
          <xs:element name="desc" minOccurs="0" />
          <xs:element name="da" type="xs:IDRef" minOccurs="0" />
          <xs:element name="ra" type="xs:IDRef" minOccurs="0" />
          <xs:element name="src" type="xs:IDRef" minOccurs="0" />
          <xs:element name="asc" type="xs:IDRef" minOccurs="0" />
          <xs:element name="dst" type="xs:IDRef" minOccurs="0" />
        </xs:all>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Figure 4-7: The TRM-XML Schema listing

Finally <*TRM*> is the single root element that includes all of <*node*> elements. This element opens once at the start of file and closes at the end.

4.3.2 Example 1

As an example, suppose one needs to express these statements: Tim is a lecturer, Java is a course, and Tim teaches Java. Here, the mentioned TRM schema can be used to structure the TRM-XML data listed in Figure 4-8.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <TRM xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
    xsi:noNamespaceSchemaLocation="C:\Research\TRMXML\TRM.xsd">
  - <node>
      <id>B01</id>
      <desc>Tim</desc>
    </node>
  - <node>
      <id>B02</id>
      <desc>Java</desc>
    </node>
  - <node>
      <id>C01</id>
      <desc>Lecturer</desc>
    </node>
  - <node>
      <id>C02</id>
      <desc>Course</desc>
    </node>
  - <node>
      <id>P01</id>
      <desc>Teaching</desc>
      <da>teaches</da>
      <ra>is taught by</ra>
    </node>
  - <node>
      <id>P02</id>
      <da>is a</da>
      <ra>is the type of</ra>
    </node>
  - <node>
      <id>S01</id>
      <src>B01</src>
      <asc>P02</asc>
      <dst>C01</dst>
    </node>
  - <node>
      <id>S02</id>
      <src>B02</src>
      <asc>P02</asc>
      <dst>C02</dst>
    </node>
  - <node>
      <id>S03</id>
      <src>B01</src>
      <asc>P01</asc>
      <dst>B02</dst>
    </node>
  </TRM>
```

Figure 4-8: The TRM-XML listing of example 1

Because of the TRM's bi-directionality, the above set of the TRM relations can express the following relations in the same time: Lecturer is the type of Tim, Course is the type of Java, and Java is taught by Tim.

Also one can build another relation about a relation. For example, to say that "Amir knows that Tim teaches Java" it is enough to add the following lines:

```
<node>
	<id>B03</id.>
	<desc>Amir</desc>
</node>
<node>
	<id>P03</id>
	<desc>knowing</desc>
	<da>knows</da><ra>is known by</ra>
</node>
<node>
	<id>S04</id>
	<src>B03</src><asc>P03</asc><dst>S03</dst>
</node>
```

Which again, at the same time expresses that "Tim teaches Java is known by Amir" or "Java is taught by Tim is known by Amir" or "Amir knows that Java is taught by Tim", etc. The above interpretations of a single fact may seem obvious for human reading, but not for the computers. This shows how bi-directionality can expands the expressed meanings when the number of interconnected the TRM relations increase. The application of this multiple-interpretation would be more flexibility in database querying.

Now if one wants to assign a course code (say "C001") to that Java course, the following lines must be added:

```
<node>
        <id>B04</id>
        <desc>C001</desc>
</node>
<node>
        <id>P04</id>
        <da>is the code of</da>
</node>
<node>
        <id>S05</id>
        <src>B04</src><asc>P04</asc><dst>B04</dst>
</node>
```

More importantly, if one wants to use "teaching" as object or subject of a statement, like to say that "Tim likes teaching", the following lines must be added.

```
<node>
        <id>P05</id>
        <da>likes</da>
</node>
<node>
        <id>S06</id>
        <src>B01</src><asc>P05</asc><dst>P01</dst>
</node>
```

The reuse of P01 as object without redefining it as a separate node is noticeable.

4.3.3 Example 2

Recalling the sample database of section 2.1.2 and the relevant TRM graph shown in Figure 3-3, writing the TRM-XML list is a straightforward process. The result is the following list:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TRM xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="TRM.xsd">
<node><id>IDTitle</id>
   <desc>Title</desc>
   <da>is titles as</da><ra>is the title of</ra>
</node>
<node><id>IDAuthor</id>
   <desc>Author</desc>
   <da>is written by</da><ra>is the author of</ra>
</node>
<node><id>IDYear</id>
   <desc>Year</desc>
   <da>is published in</da><ra>is the year of publication of</ra>
</node>
<node><id>IDJournal</id>
   <desc>Journal</desc>
   <da>is published by</da><ra>is the publisher of</ra>
</node>
<node><id>IDAuthor1</id>
   <desc>C. Bussler</desc>
</node>
<node><id>IDTitle1</id>
   <desc>Enterprise-Wide Workflow Management</desc>
</node>
<node><id>IDYear1</id>
   <desc>1999</desc>
</node>
<node><id>IDJournal1</id>
   <desc>IEEE Concurrency</desc>
</node>
<node><id>IDArticle1</id>
   <desc>Article1</desc>
</node>
<node><id>IDArticle1R1</id>
   <src>IDArticle1</src><asc>IDTitle</asc><dst>IDTitle1</dst>
</node>
<node><id>IDArticle1R2</id>
```

```
    <src>IDArticle1</src><asc>IDAuthor</asc><dst>IDAuthor1</dst>
</node>
<node><id>IDArticle1R3</id>
    <src>IDArticle1</src><asc>IDYear</asc><dst>IDYear1</dst>
</node>
<node><id>IDArticle1R4</id>
    <src>IDArticle1</src><asc>IDJournal</asc><dst>IDJournal1</dst>
</node>
<node><id>IDArticle2</id>
    <desc>Article2</desc>
</node>
<node><id>IDAuthor2a</id>
    <desc>S. Choenni</desc>
</node>
<node><id>IDAuthor2b</id>
    <desc>R. Bakker</desc>
</node>
<node><id>IDTitle2</id>
    <desc>On the Evaluation of Workflow Systems in Business Processes</desc>
</node>
<node><id>IDYear2</id>
    <desc>2003</desc>
</node>
<node><id>IDJournal2</id>
    <desc>Electronic Journal of Information Systems Evaluation</desc>
</node>
<node><id>IDArticle2R1</id>
    <src>IDArticle2</src><asc>IDTitle</asc><dst>IDTitle2</dst>
</node>
<node><id>IDArticle2R2</id>
    <src>IDArticle2</src><asc>IDAuthor</asc>
    <dst>IDAuthor2a</dst>
</node>
<node><id>IDArticle2R3</id>
    <src>IDArticle2</src><asc>IDAuthor</asc>
    <dst>IDAuthor2b</dst>
</node>
<node><id>IDArticle2R5</id>
```

```
    <src>IDArticle2</src><asc>IDJournal</asc><dst>IDJournal2</dst>
</node>
<node><id>IDArticle2R4</id>
    <src>IDArticle2</src><asc>IDYear</asc><dst>IDYear2</dst>
</node>
<node><id>IDArticle3</id>
    <desc>Article3</desc>
</node>
<node><id>IDYear3</id>
    <desc>2006</desc>
</node>
<node><id>IDTitle3</id>
    <desc>Searching for e-Business Performance Measurement Systems</desc>
</node>
<node><id>IDArticle3R1</id>
    <src>IDArticle3</src><asc>IDTitle</asc><dst>IDTitle3</dst>
</node>
<node><id>IDArticle3R2</id>
    <src>IDArticle3</src><asc>IDJournal</asc><dst>IDJournal2</dst>
</node>
<node><id>IDArticle3R3</id>
    <src>IDArticle3</src><asc>IDYear</asc><dst>IDYear3</dst>
</node>
</TRM>
```

4.4 Discussion

After introducing the TRM-DB, it is necessary to notice that XML and tables may be viewed as "mechanics" of expressing the information modelled in the TRM. The TRM-DB in the data model layer uses the TRM as the foundation layer and the TRM-XML or the TRM-Table as data storage layer. The point that makes the TRM-DB special is that unlike other related works in the data model layer, this database is directly based on the TRM. For example, ZigZag is also an information layer based on the Static-TRM, but it doesn't use all of what the TRM can provide.

The TRM-DB can be theoretically used to describe many structured or unstructured real-world information with integration of schema in the database. This characteristic is based on describing all information, whether data or metadata with the same method and in a same context. The main issue is that there is a single, simple and global schema which is not dedicated to any particular database.

As a comparison between the studied data models, the TRM-DB has disadvantages and advantages over RDB, XML and Zzstructure, which have been summarized in Table 4-1.

Table 4-1: Comparison of features the studied data models

	RDB	XML	Zzstructure	TRM-DB
Meta-data	Separated	Joint	No or Mixed	No or Mixed
One-to-many relationship	Yes	Yes by repetition	Yes by transclusion	Yes
Null problem	No by decomposition (if practical)	No	No	No for TRM-XML, Yes for TRM-Table
Re-use	Yes by normalization	Yes by ID/IDREF	Built-in	Built-in
Hierarchy	Yes	Yes	No	No
Bi-directionality	No	No	No	Yes
Relative required storage space	Low	High	High	High

4.5 Querying the TRM-DB

Querying the TRM-DB is different in nature from querying other databases like RDBs or XML. The difference goes back to the lack of a schema in the TRM-DB. For example, a question like "what are the titles of the articles published in year 2003" in RDBs is convertible to a SQL statement having "…WHERE year=2003…". In that statement, a part of the schema ("year") is questioned to be equal to a value ("2003"). Similarly in XML, an element called "year" may be examined to be equal to that value in an XQuery statement.

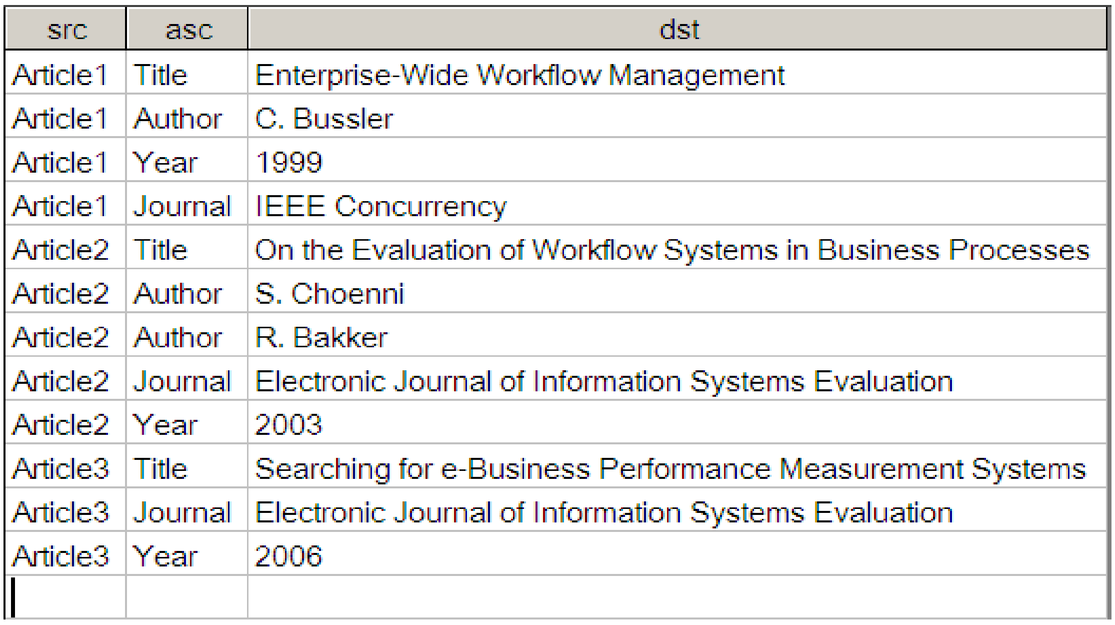
In the TRM-DB by contrast, some strings like “year” are parts of data, like any other data like “2003”. The only questionable things in SQL or XQuery are id, da, ra, src, asc and dst, so a mapping method between these two kinds of queries must be developed. In this section, the mapping method is described through two examples for the TRM-Table and the TRM-XML. The used database is the example of bibliographic data of section 2.1.2.

4.5.1 Querying the TRM-Table

The equivalent TRM-Table (called “Nodes”) has been shown in Figure 4-6. For querying the TRM-table, first a view is designed called RelationTriples, as follows:

```
CREATE VIEW RelationTriples
SELECT Nodes_1.desc AS src, Nodes_2.desc AS asc, Nodes_3.desc AS dst
FROM ((Nodes
INNER JOIN Nodes AS Nodes_1 ON Nodes.src=Nodes_1.ID)
INNER JOIN Nodes AS Nodes_2 ON Nodes.asc=Nodes_2.ID)
INNER JOIN Nodes AS Nodes_3 ON Nodes.dst=Nodes_3.ID;
ORDER BY src;
```

This view provides two features, first it filters the database with the records which have completed triples of src, asc and dst (means relations only), secondly it shows the TRM representation between real pieces of data, not between their identifiers. This view may contain redundancies, but not any null. The sample output is shown in Figure 4-9.



src	asc	dst
Article1	Title	Enterprise-Wide Workflow Management
Article1	Author	C. Bussler
Article1	Year	1999
Article1	Journal	IEEE Concurrency
Article2	Title	On the Evaluation of Workflow Systems in Business Processes
Article2	Author	S. Choenni
Article2	Author	R. Bakker
Article2	Journal	Electronic Journal of Information Systems Evaluation
Article2	Year	2003
Article3	Title	Searching for e-Business Performance Measurement Systems
Article3	Journal	Electronic Journal of Information Systems Evaluation
Article3	Year	2006

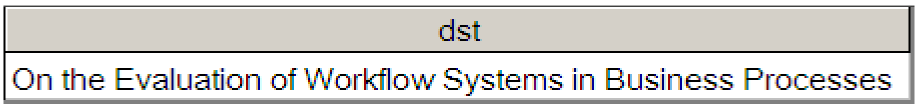
Figure 4-9: The view of "RelationTriples" applied on the sample database

Then RelationTriples view can be used for applying queries through some stages. For example, to query “what are the titles of the articles with year=2003”, first one must filter on (asc=”Year” AND dst=”2003”) to find src as “Article2”, then another filter must be (src=”Article1” AND asc=”title”) to find dst as “On the Evaluation of …”. This can be done in SQL using a single statement like:

```
SELECT RelationTriples_2.dst
FROM RelationTriples AS RelationTriples_1
INNER JOIN RelationTriples AS RelationTriples_2
ON RelationTriples_1.src = RelationTriples_2.src
WHERE RelationTriples_1.asc="year"
AND RelationTriples_1.dst="2003"
AND RelationTriples_2.asc="title";
```

The above statement relates RelationTriples to itself, with aliases “RelationTriples_1” and “RelationTriples_2”. Then the condition “RelationTriples_1.asc=year AND RelationTriples_1.dst=2003” will filter the records to the wanted relations, with their src referring to the wanted articles. For the wanted articles, their titles are required. The relations that can tell us the titles are those with “title” in their asc field. Thus in RelationTriples_2 a relation with asc=title and src={what is already found}is required. Also RelationTriples_2 has been already filtered on its src (because it joins to RelationTriples_1 by common src’s).

The result of running the query is shown in Figure 4-10.



dst
On the Evaluation of Workflow Systems in Business Processes

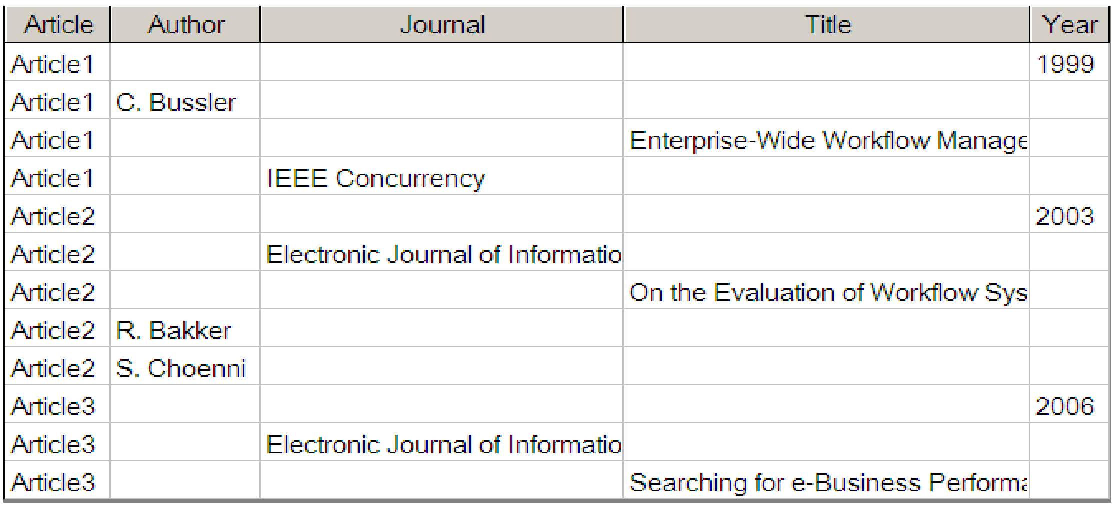
Figure 4-10: The sample output of querying the TRM-Table

4.5.1.1 Rebuilding the Relational Database

If a database is designed in the TRM-Table, it can be used to build the equivalent relational database. An SQL statement can use the designed “RelationTriples” view, and give a full non-normalized version of the database, i.e. a big table with all the possible columns. The required SQL statement uses TRANSFORM and PIVOT keyword to convert data from cells to the headers of the columns, as follows:

```
TRANSFORM First(RelationTriples.dst) AS FirstOfdsc
SELECT RelationTriples.src AS Article
FROM RelationTriples
GROUP BY RelationTriples.src, RelationTriples.dst
PIVOT RelationTriples.asc;
```

The result is like Figure 4-11.

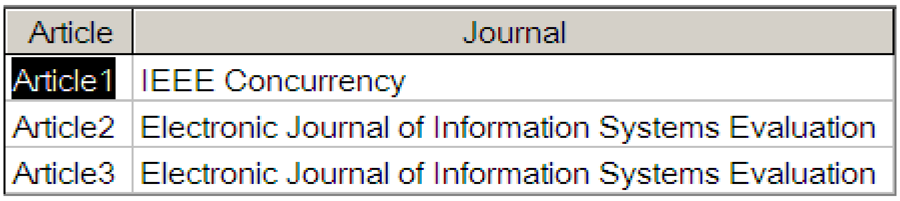


Article	Author	Journal	Title	Year
Article1				1999
Article1	C. Bussler			
Article1			Enterprise-Wide Workflow Manage	
Article1		IEEE Concurrency		
Article2				2003
Article2		Electronic Journal of Informatio		
Article2			On the Evaluation of Workflow Sys	
Article2	R. Bakker			
Article2	S. Choenni			
Article3				2006
Article3		Electronic Journal of Informatio		
Article3			Searching for e-Business Performa	

Figure 4-11: The sample result of converting the TRM-Table to full non-normalized table

The resulted table implicitly shows the binary decomposed version of the relational database. Unlike normal progress, now it can be used to "compose" the required relations. As the first step, one can remove null values and end up with individual binary tables (but with possible redundancies) by querying the above SQL statement for some specific asc. For example, the following statement will give the result of Figure 4-12

```
TRANSFORM First (RelationTriples.dst) AS FirstOfdsc
SELECT RelationTriples.src AS Article
FROM RelationTriples
WHERE RelationTriples.asc="journal"
GROUP BY RelationTriples.src, RelationTriples.dst
PIVOT RelationTriples.asc;
```



Article	Journal
Article1	IEEE Concurrency
Article2	Electronic Journal of Information Systems Evaluation
Article3	Electronic Journal of Information Systems Evaluation

Figure 4-12: A binary redundant table produced by transforming the TRM-Table

Finally if identifiers are used, the result is a non-redundant set of the fully-decomposed tables.

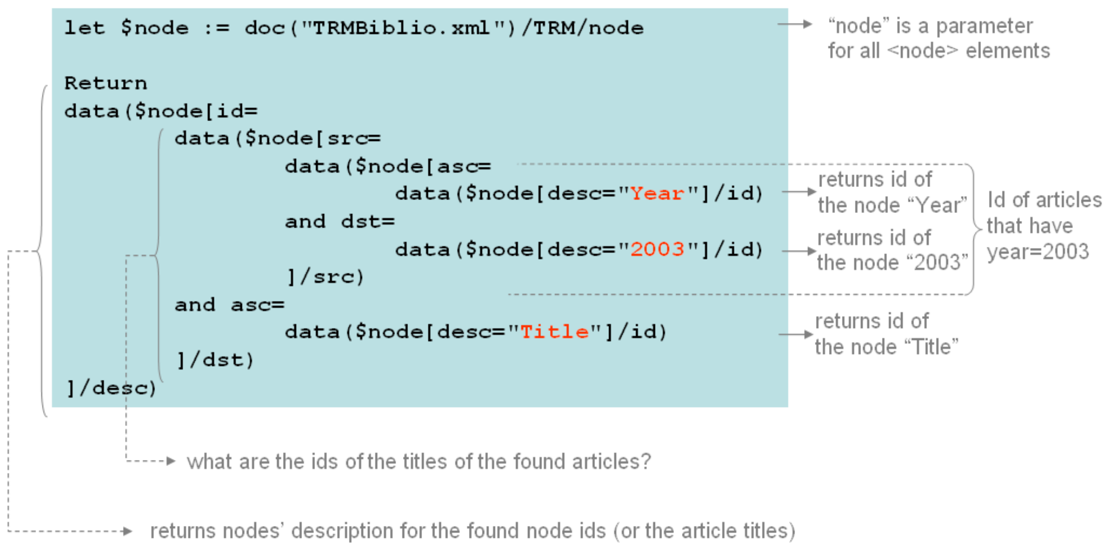
4.5.2 Querying the TRM-XML

XQuery is a language developed for querying XML databases. It has almost the same role for XML as SQL has for RDBs. XQuery is the recommended query language for XML by W3C (http://www.w3.org) and has the potential to be one of the most important query languages [48]. However, XQuery is not yet supported in any Web browser at the time of writing this thesis. Rather, there are many implementations of it in terms of applications, plug-ins or as a part of some database engines. More about the syntax of XQuery is outside the scope of this thesis and the full documentations can be found on W3C website (http://www.w3.org).

XQuery can be used as a query language for the TRM-DB because it can query the TRM-XML. The way XQuery is used for querying the TRM-XML is again naturally different from the way it is used to query any other XML file. Here XQuery can only question about special elements (id, da, ra, src, asc, dst) and the questions must be mapped to that special way of using XQuery. This has been explained in the following example.

Recalling the bibliographic example and the equivalent TRM-XML listing shown in section 4.3.3, let us suppose the query is again to find "What are the titles of the articles published in year 2003". The required XQuery is shown and explained in Figure 4-13.

Query: What are the titles of articles published in year 2003?



```
let $node := doc("TRMBiblio.xml")/TRM/node

Return
data($node[id=
        data($node[src=
                data($node[asc=
                        data($node[desc="Year"]/id)
                and dst=
                        data($node[desc="2003"]/id)
                ]/src)
        and asc=
                data($node[desc="Title"]/id)
        ]/dst)
]/desc)
```

"node" is a parameter for all <node> elements
returns id of the node "Year"
returns id of the node "2003"
Id of articles that have year=2003
returns id of the node "Title"
what are the ids of the titles of the found articles?
returns nodes' description for the found node ids (or the article titles)

Output for the sample database:
"On the Evaluation of Workflow Systems in Business Processes"

Figure 4-13: Description of a sample the TRM query statement in XQuery.

4.6 Summary

In this chapter, the TRM-DB has been introduced as a method of data structuring within a schemaless framework. It is an approach of structuring information directly on top of Static-TRM layer. Two implementations are proposed to make the TRM-DB feasible: The TRM-Table which uses a single table to store the entire database, and the TRM-XML which uses XML as a language of serializing the TRM nodes.

Before introducing the TRM-DB, the graphical notation of the TRM has been shown to be able to express RDBs, ZigZag and XML. Finally the methods of querying the TRM-DB are provided, having two implementations: Using SQL for the TRM-Table and XQuery for the TRM-XML.

Chapter 5 -

TRM-NAV: A NEW HYPERTEXT NAVIGATION MODEL

Recalling from section 2.4, while classic hypertext models (like the Dexter Model [92]) define a link as a two-element object consisting source and destination, 'association' has sometimes been considered as the third element, varying from implicit to explicit involvement. Such three-element links has been called "Ternary-Links" (as an extension to the binary links of the BRM [19]) hereafter.

This chapter explains a TRM-based navigation model of hypertext, called "TRM-NAV". Like the previous chapters, this model has a unifying approach and tries to make a general framework to cover all implicit and explicit approaches to the concept of the Ternary Links.

5.1 Background

When the BRM [19] was introduced in chapter 2, a ternary approach to the BRM in section 2.5.2 showed the limitations of the BRM in modelling the navigation in a class of hypertext systems called "knowledge-oriented hypertext". The BRM focused on the four main navigational questions (Is this node a link source? Where can I go from this source? Is this node a link destination? What nodes are linked to this destination?). However, it represents explicitly nothing else, such as the semantics or types or meaning of links. The extra information about a binary link may be implicitly presented in the implementation. For example, inspection of a set of links or of a process generating links may indicate its purpose; however, this is not expressed explicitly in the model.

What is semantically lacking from the BRM representations is a discriminator which encapsulates information about the semantics of the relation, such as the relation's name, type, purpose or selection criteria. While this information is not essential for representing relations, it is however useful in varying degrees in different real-world applications. An obvious example is in hypertext itself, where a chooser function can be most helpful when numerous possible links are available from a given source. In another application, it might be useful to perform calculations which take into account the semantics of the relations and relation incidences, for example, finding all elements which satisfy the relation "is-a-kind-of tree" but not "lives-in tree".

The BRM represents no information that would assist a user in selecting the most appropriate link destination for their purposes. In the BRM, the navigational questions allow one to determine where one can go and from where, but not *why* one would want to go there.

5.1.1 Extending the BRM

The concept of the TRM may be used to extend the BRM navigational model (described in section 2.5) minding the described semantic limitations. A third feature of a relation is introduced whose purpose is to explicitly represent the semantics of the relation that binds elements. So for example, the BRM's pE representation (section 2.5.1), specified by a predicate p and expression E is now supplemented with a designator f which can be a relation name or any other semantically-meaningful description, and now becomes a *pfE* relation. Also the "functional links" of BRM become *semantic functional links*. The BRM enumeration of incidences representation which represents a binary relation as a set of pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ now becomes $\{(x_1, R, y_1), (x_2, R, y_2), ..., (x_n, R, y_n)\}$ where R is the relation to which these pairs all belong. This follows the Static-TRM formulation of section 3.1.1.

The pfE linking as the TRM equivalent of the BRM's pE linking, by analogy is the most general case when the relation is represented as a *(predicate, expression, expression)*, or when $R=\{(x,a,y) \mid p(x)=true\ ,\ a=f(x)\ ;\ y=g(x,a)\}$. This follows what has been formulated as the Dynamic-TRM in section 3.1.2. The above is a general case when an unadvertised source of a link can be filtered by the *p()* function, then the available associations are computed by knowing the source, and finally the available destinations are computed by knowing the source and the nominated associations.

Recalling the example of pE links about journal article links on section 2.5.1, an example of pfE link could be a page similar to that example, having ternary links. For example if a journal title on the hypertext page is qualified to be a link source (using the Predicate mechanism) then one or more associations are calculated using the f() function. This function can return terms like "Abstract", "Authors", etc. which can be a variable list for different users. After the user selects a certain association, then the destination is again calculated using another function E(). The function E() may use the result(s) of function f() as parameter, as well as user's level of access, etc. to determine the page that the user will be taken to.

This case will be explained in more detail in section 5.8.3.

5.2 The TRM-NAV Definition

Following the described principles for the BRM extension, "TRM-NAV" is introduced to be a "navigational model" for hypertext systems based on the TRM (ranging from the Static-TRM to the Dynamic-TRM), when:

1- The TRM's source node is the source endpoint in the links of a hypertext system.

2- The TRM's destination node is the destination endpoint in the links of a hypertext systems

3- The TRM's association node is a hypertext resource that describes the relation between the source and the destination.

This general approach provides an integrated framework for a number of approaches to navigational modelling, and as such it is likely to be a valuable technique to use in future designs of hypertext infrastructure.

5.2.1 The TRM-NAV Navigational Questions

By analogy, if the 2 elements of the BRM result in 4 questions (in section 2.5.1), then 3 elements of the TRM-NAV can provisionally result the 9 following questions:

1- Outgoing Questions:

   a. Is this node (*x*) the source of any link?

   $\exists (x,*,*) \in R$

   b. What are the associations (*a*) of link originated by this node?

   $\{a \in S \mid (x,a,*) \in R\}$

   c. Where can I go starting with this node via any found association?

   $\{y \in S \mid (x,a,y) \in R\}$

2- Incoming Questions:

   a. Is this node (*y*) the destination of any link?

   $\exists (*,*,y) \in R$

   b. What are the associations (*a*) of links terminating with this node?

   $\{a \in S \mid (*,a,y) \in R\}$

   c. What nodes are linked to this node via any found association?

   $\{y \in S \mid (x,a,y) \in R\}$

3- Associative Questions:

   a. Is this node (*a*) the association of any link?

   $\exists (*,a,*) \in R$

   b. What are the sources (*x*) of those links?

   $\{x \in S \mid (x,a,*) \in R\}$

   c. What are the destinations of those links for each found source?

   $\{y \in S \mid (x,a,y) \in R\}$

The above questions are built by analogy to the BRM, but they can be simplified because the process of finding the second and the third element of each link are almost mixed and the second and the third question in each group can be combined. The reduced set contains 6 questions as follows:

1- *Source Existence:* Is this node (*x*) the source of any link?

$\exists (x,*,*) \in R$

2- *Outgoing Links Identification:* What are the links originated by this node?

$\{(x,a,y) \in R\}$

3- *Destination Existence:* Is this node (*y*) the destination of any link?

$\exists (*,*,y) \in R$

4- *Incoming Links Identification:* What are the links terminating with this node?

$\{(x,a,y) \in R\}$

5- *Association Existence:* Is this node (*a*) the association of any link?

$\exists\ (*,a,*) \in R$

6- *Links Identification:* What are the existing links via this association?

$\{(x,a,y) \in R\}$

5.2.2 Link Implementations

As with the conclusions drawn from the BRM, many implementations of hypertext systems are based on mixing the above TRM-NAV questions with the static and dynamic states of endpoint definitions. The difference however, is that having triples such as *(x,a,y)* implies that each element may be *enumerated*, *predicate* or *expression*. Some of the important possible implementations are:

(enumerated, enumerated, enumerated)

$R=\{(x,a,y)\ |\ \textit{some}\ x,a,y \in S\}$

This is the formation common to hypertext systems that explicitly contain relation triples, such as zzstructure and the Semantic Web. Indeed the World-Wide Web may be viewed as a special case of this, where one of the enumerations is a single arbitrary constant.

(enumerated, enumerated, expression)

$R=\{(x,a,y)\ |\ \textit{some}\ x,a \in S\ , y=f(x,a)\}$:

This is an interesting case, where a destination may be computed from knowing where one is and how one wants to navigate away from this starting point.

(enumerated, expression, expression)

$R=\{(x,a,y)\ |\ \textit{some}\ x \in S\ , a=f(x), y=g(x,a)\}$

Another interesting case, when sources are explicitly defined before, but selecting a link source can compute or list all of the traceable association, then selecting a particular association can compute or list all of the possible destinations. The created links by Dynamic Hypertext Engine (DHE) [29, 30] studied in section 2.4.5 is an example of this case. DHE applies Relationship Navigation Analysis (RNA) on a relational database contents and create new dynamic links on the content of an existing hypertext. In such created links, x is the content of a field in a database table, a is an available association originating from x and computed by RNA and y is the link endpoint computed by knowing x and a.

(expression, enumerated, enumerated)

$R=\{(x,a,y) \mid \textit{some } a,y \in S,\ x=f(y,a)\}$

In this case for the pre-computed links, knowing where the user is currently in and she/he could have gone to this place can lead to all the sources where she/he could have been be before. When links are dynamically computed, this means that knowing where and how one wants to go afterward, can make the current element an unadvertised hyperlink source.

(expression, expression, enumerated)

$R=\{(x,a,y) \mid \textit{some } y \in S,\ a=f(y)\,;\ x=g(y,a)\}$

Considering where the user is, the system can compute the sources where he/she could have been before and how each source can be linked to this destination.

(predicate, enumerated, expression)

$R=\{(x,a,y) \mid \textit{some } a \in S,\ p(x)=\textit{true}\,;\ y=g(a)\}$

This case is where the link endpoints are defined dynamically, i.e. the unadvertised source and destination.

(predicate, expression, expression)

$R=\{(x,a,y) \mid p(x)=\textit{true},\ a=f(x)\,;\ y=g(x,a)\}$

This is the general case which was called pfE or the semantic functional link. In this case an unadvertised source of a link can be filtered by a function, then the available associations are computed by knowing the source, and finally the available destinations are computed by knowing the source and the nominated associations.

Where accuracy is critical, it is possible that the predicate *p()* would actually be a Boolean conjunction of the results of the function *f()*, so that if at least one relation type is found for a given potential source, then the predicate returns true, i.e. that the potential source is indeed a source, because an association (link type) was found for it.

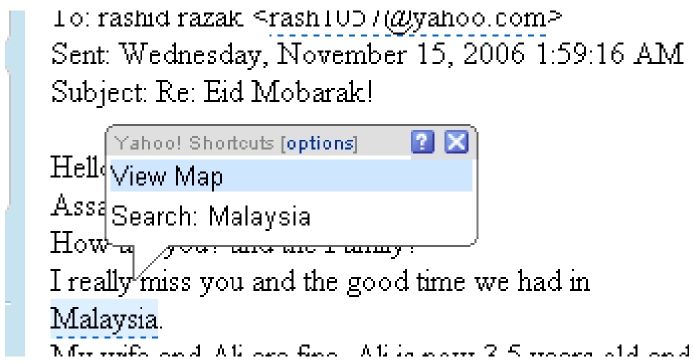
5.3 The Ternary-Links

The TRM-NAV is based on three elements for hypertext links. Although the familiar binary links of hypertext can be modelled in the TRM-NAV, a more general approach in the implementation of hypertext system is to bring that ternary principle to the user's vision. A Ternary Link is a hypertext link for which the user can explicitly see the association of the link as a separated piece of information. Moreover, some information about the destination can also be shown to the user inside the same Ternary Link. It is important to notice that the Ternary-Links are not new and the TRM-NAV main contribution is not to show new implementations of links, but it is to show a general modelling framework for many implemented or unimplemented works. This section explains some of the related works about the implemented Ternary-Links.

5.3.1 Examples

There are some recent developments in the enrichment of the Web's linking mechanism through visualization tools, which can be studied to support the concept of the Ternary Links. Two found examples of these developments (out of many similar systems) are *Yahoo Shortcuts* (http://tools.search.yahoo.com/shortcuts/) and *Snap Shots* (http://www.snap.com/about/shots.php). From the point of view of this thesis, they may be considered as adding a third dimension to traditional Web links.

In "Yahoo Shortcuts", the body text of an email in the Yahoo environment is searched automatically to uncover potential link anchors, and potential anchors are distinguished by the addition of dashed underlines to the corresponding text. By hovering with the mouse pointer over these special links, a menu of choices appears that is dependent on the nature of that specific anchor. The user is then directed to some other Web pages by selecting an appropriate menu choice. A Boolean function determines whether or not each word should be considered as an anchor. Then this information is fed into a menu-producing function. The options of this menu (equivalents to associations) and the destination Web page for each one are also either pre-computed or dynamically computed. An illustration of the user interface is shown in Figure 5-1.

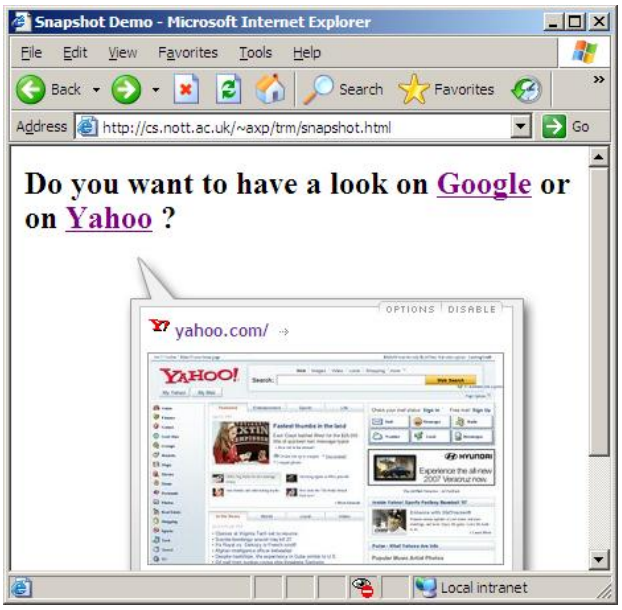


To: rashid razak <rash1057@yahoo.com>
Sent: Wednesday, November 15, 2006 1:59:16 AM
Subject: Re: Eid Mobarak!

Hell
Assa
How
I really miss you and the good time we had in
Malaysia.
My wife and Ali are fine. Ali is now 3.5 years old and

Figure 5-1: A sample Yahoo Shortcut screen capture

"Snap Shots" provides capabilities to preview a destination web page before the user actually clicks on the link anchor. This works by showing a small window when the user hovers the mouse cursor over the link anchor. Snapshot can be considered as an example of using Ternary Links in the Web. From this thesis's point of view, the content of the small preview window is in fact a visual association between the link anchor and the destination page. A sample screen capture of Snap Shots is shown in Figure 5-2.



Snapshot Demo - Microsoft Internet Explorer

Address http://cs.nott.ac.uk/~axp/trm/snapshot.html

Do you want to have a look on Google or on Yahoo ?

Figure 5-2: A sample Snap Shot screen capture

5.4 Discussion

The TRM-NAV offers a number of benefits as a navigational model. It supports knowledge-orientation, and is an open approach because the links are abstracted from the nodes. Link associations, as some independent information units, can participate in other links as association, source or destination, facilitating the reuse of link structures. Link associations are flexible, with their own structure and management, and bi-directional. Potential problems with this model are that it increases complexity of the stored information (because links are based on three rather than two elements), and backwards compatibility is problematic (both in respect of the third element of links and in their bi-directionality).

There are potential misunderstandings about the TRM-NAV that shall be addressed here. The first is the misconception that a TRM's triple of (x,a,y) can be decomposed into two BRM's couples of (x,a) and (a,y). It is possible to mistakenly conclude that the TRM-NAV is nothing more than a reconfigured BRM. For example, it may be presumed that even if links have explicit associations, the user first selects x as the source and goes to a as an intermediate point, then selects a as the source and goes to y as destination. This view is flawed because the decomposition is not retractable. Decomposing (x,a,y) into two couples will lose the logical relation between x and y, which was the origination for establishing the link. Formally, there will be no way to discovering the relation between x_1 and y_1 while having a set of couples like $\{(x_1,a),(x_2,a),(a,y_1),(a,y_2)\}$.

Another issue is to ask the question that if the BRM can be extended to the TRM, may it be extended to upper degrees like 4RM, 5RM, or higher? This is equivalent to ask why a link must only have one internal attribute that is considered as association. The reason for this is that the link association should not be viewed as an attribute of the link or link set, and in fact this is the reason that an association is not necessarily a synonym for "link type". Instead, it is a node that contains all of the informational content for expressing a relation between two other nodes.

5.5 An Implemented Demonstration

Although Ternary Linking is a general method and hence may be implemented in a variety of different situations, it will be illustrated here by using a simple Ternary Link implementation.

A hypertext system is designed that uses explicit Ternary Links for navigation, where the central tenet is to add some Ternary Links to the existing web pages using an abstracted ternary linkbases. In principle, Ternary Links could be added by a wide variety of mechanisms - encompassing both human authoring and link computation.

Each linkbase consists of a set of facts expressed as triples. Thus an existing Web page may be enriched when viewed in different contexts*,* by applying different Ternary Links that may come from different domains. The user interface consists of a drop-down menu containing hyperlinked items that appears when a user hovers the mouse cursor over the link anchor. The system allows the user to follow links in the binary mode as well, by simply clicking on them as usual. Although the idea of adding semantically rich links in this way is not in itself new, here links that are based upon ternary facts are being added.

The system uses the following technologies: 1) MySQL to manage the information about the nodes and the Ternary Links, 2) PHP to access the MySQL database and to modify the existing web pages, and 3) JavaScript and CSS to manage the new user interface of the added links. The programming codes are listed in Appendix C.

The designated PHP script receives a URL as a parameter and utilises this to produce the modified web page. The system searches the body text for any occurrence of the "description" field of the "Node" table, and then for each found node it searches the "Link" table for any matches in "source" or "destination" fields. The "direct association" field for each matched "source" and the "reverse association" field for each matched "destination" will be found in the "Node" table - their URLs will produce items of the drop-down menu for the candidate nodes.

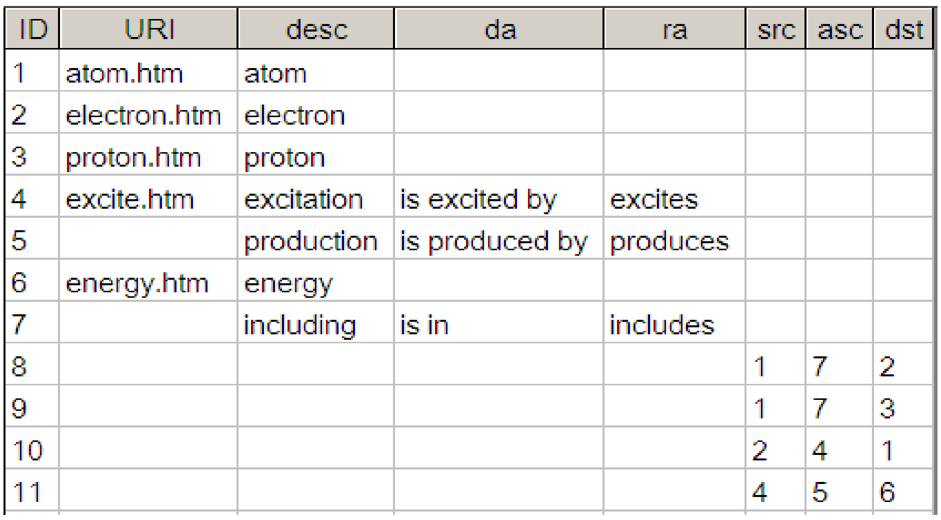
5.5.1 *An Example*

Supposing there are web pages describing physical concepts for: Atom(http://www.myweb.com/atom.html), Electron(http://www.myweb.com/electron.html), Proton(http://www.myweb.com/proton.html), Neutron(http://www.myweb.com/neutron.html), Excitation(http://www.myweb.com/excitation.html).

The author of these web pages wishes to express these physical facts "through user's navigation":

1- Atom includes protons and electrons.
2- Electrons are able to excite atoms.
3- Excitation produces energy

According to chapter 4, the TRM-Table is selected to represent the above information; however, the TRM-XML can also be another choice. Figure 5-3 shows the TRM-Table produced for the above facts.



ID	URI	desc	da	ra	src	asc	dst
1	atom.htm	atom					
2	electron.htm	electron					
3	proton.htm	proton					
4	excite.htm	excitation	is excited by	excites			
5		production	is produced by	produces			
6	energy.htm	energy					
7		including	is in	includes			
8					1	7	2
9					1	7	3
10					2	4	1
11					4	5	6

Figure 5-3: the TRM-Table equivalent for Atom Page example

Using the id's specified in the TRM-Table, a TRM graph can be drawn to show the ternary relations between the nodes. This is shown in Figure 5-6.

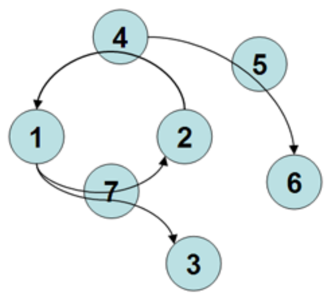
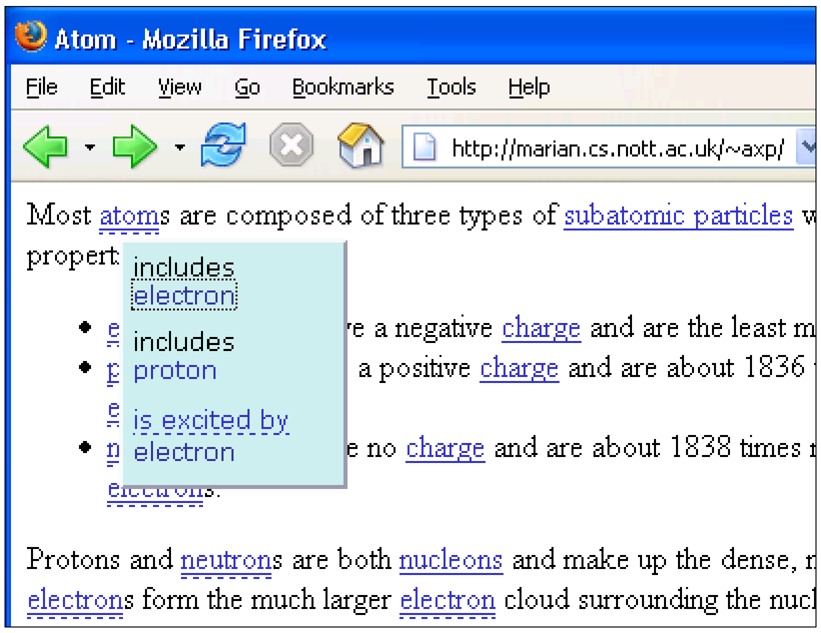


Figure 5-4: the TRM graph of the Atom Webpage example

After executing the system, a sample screenshot of the produced Ternary-Links are shown in Figure 5-5. This has also been demonstrated in http://cs.nott.ac.uk/~axp/trm.



Atom - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://marian.cs.nott.ac.uk/~axp/

Most atoms are composed of three types of subatomic particles w
propert

- e ve a negative charge and are the least m
- p a positive charge and are about 1836
  e
- n e no charge and are about 1838 times
  electrons.

includes electron

includes proton

is excited by electron

Protons and neutrons are both nucleons and make up the dense, r
electrons form the much larger electron cloud surrounding the nucl

Figure 5-5: Screenshot of a prototype implementation of a Webpage with Ternary-Link

Although this example is based on single-word concepts, the approach can be used for building relations between more substantial items of information in a broader view. Also this approach can readily be extended to different linkbases, permitting the production of different sets of Ternary Links on the same web page.

The importance of this example is that here, the TRM is not used to express statements only, but the main purpose is to express the meanings of links between several web pages. Also this example can clearly show why the TRM-NAV is different from the philosophy of the Semantic Web. Here the TRM-NAV has targeted humans, not machines. Machines are not interested to know about the sources, association and destination of links, because they won't traverse links to know more. This is while the philosophy of the TRM-NAV is transferring knowledge through navigation of links. For example, users will understand two different relations between atom and electron by following two different links, in which the only difference is in their association. The transferred knowledge is more than two short statements described by the equivalent RDF triples. The flexibility of the link traversal, the user choice to select the desired link destination, and the whole environment can totally transfer the knowledge between the authors and the readers.

5.6 Ternary Links for Adaptive Hypermedia

In the previous section, Ternary Links were based on the Static-TRM. This section studies the potentials of the Dynamic-TRM-NAV to be applied in Adaptive Hypermedia.

5.6.1 Adaptation through Ternary Links

Hypertext systems, as mediums of knowledge transfer between authors and readers, have much potential for being adapted to the user's requirements in order to make this transfer as effective as possible. The initial definitions of the Adaptive Hypermedia [39] was "All hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various *visible* aspects of the system to the user". Also the technologies of this adaptation have been divided to "Adaptive presentation" and "Adaptive navigation". The navigational adaptation part is also typically divided to various methods which are mostly about the visual alterations: Direct guidance, sorting of links, hiding (or disabling) of links, annotation of links and map adaptation [39]. Later a new group of methods called "Adaptive Link Generation" has been added to the above categorization [37]. The nearest methods to the focus of this thesis are "direct guidance" and "link generation". The first method is altering the destination of a link in general, however this method has been mostly exemplified by "guiding" links (like next/previous buttons) [34].

The second method creates new and un-authored link (as opposed to pre-authored links) depending on the user's characteristics [38].

What a dynamic link model may do is about altering any of the two or three elements of a link. It may identify the potent source nodes and associate other nodes to that source. The system may show the same link anchor to two users, but the first user will be taken to a different destination than the second one. For example, a link on a house icon may take the buyer to the house buying page and the seller to the house selling page. In this case, the associated problem in the binary links is that the semantic of the navigation is hidden from the user unless he or she guesses the semantic after navigation, which may cause user's disorientation. Also, since both users may see the same visible interface and no visible clue tells anything about the link semantics, this would be a bit different from the Adaptive Hypermedia definition mentioned above.

Using TRM-NAV, the link adaptation can be applied on all three elements of the link, which may show a new horizon in the field of Adaptive Hypermedia. Because each Ternary Link includes not only source and destination, but also the association between these two, they help users to explicitly identify the semantic relation embedded in each adapted link. As an implementation, ternary linkbases can be used to add the required links to an established hypertext after being processed in the user's context.

5.6.2 The System Structure

A hypertext adaptation system is proposed here, which works using existing hypertext documents. A ternary semantic linkbase is used to discover potential anchors, which may form the originating point of the new links. New Ternary Links, based on the linkbase and user identification, are then computed and are finally added to the existing hypertext, together with the necessary software support to implement these new kinds of links. This process is summarised in Figure 5-6.

All the available semantic Ternary Links, which are stored in a linkbase, can be filtered depending upon the user's specification and context, and be computed and added to the set of pre-existing links in a hypertext document. The Ternary Links in the resulting document may be traversed using an embedded software.

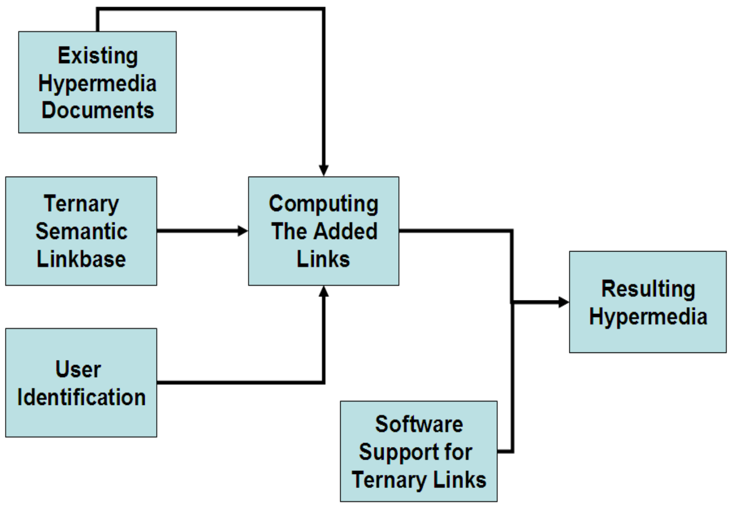

Existing Hypermedia Documents
Ternary Semantic Linkbase
Computing The Added Links
User Identification
Resulting Hypermedia
Software Support for Ternary Links

Figure 5-6: The mechanism of personalizing using added Ternary-Links.

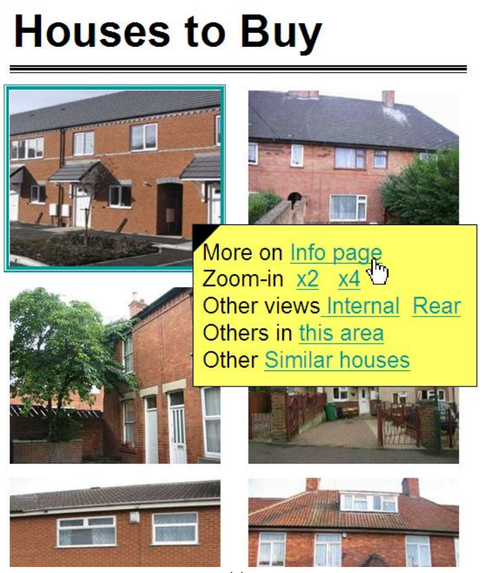
5.6.3 An Example

As an example, an online real estate photo browsing system is considered. In addition to customers that need to explore the available houses, the users of the system may include agents and administrative staff. Each of these groups may need to access to a different class of information about properties. The information has also been arranged as semantic triples (in RDF). For example, statements such as "property *x* is owned by *y*" or "property *x* is in area *z*" has been stored in the system as some triples like *(x, owned by, y)* , *(x, in area, z)*.

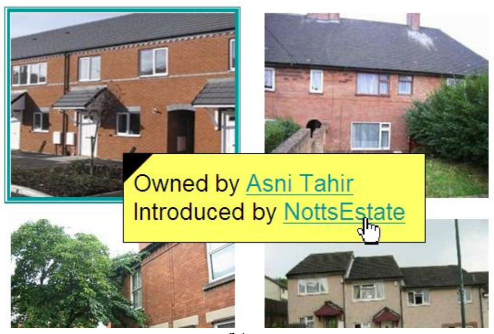
A pre-existing web page has been designed that contains the property photos. The source of the links will be computed to be the photo of a property, and the association together with the available link destination, will be displayed whenever the mouse pointer is over a specific photo. In this way, the structure of available links originating from the current object is made accessible to the user. The semantic relations between the source and the destination will be made apparent before the link is traversed.

Adaptation is implemented by the capability of making different computed Ternary Links for different users. For example the user may see links depending upon whether they are a customer (Figure 5-7-a), an agent (Figure 5-7-b) or an administrator (Figure 5-7-c).

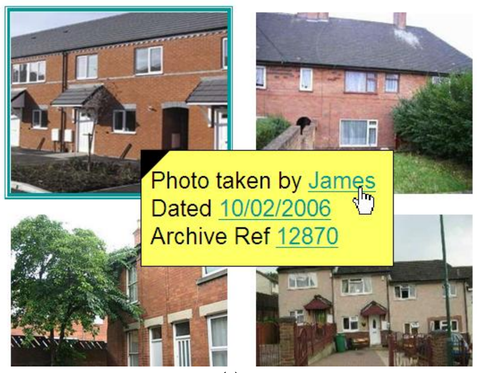
Figure 5-7(a) shows the personalized sample hypertext for user of the "customer" group.



(a)



(b)



(c)

Figure 5-7: A sample Ternary-Links example with 3 personalized looks

Figure 5-7(b) shows the personalized sample hypertext for a user of the "admin" group. In this case for example, the available Ternary Links include only those which are related to the administrative tasks. Finally Figure 5-7(c) shows the personalized sample hypertext for a user of the "agent" group.

5.7 The TRM-NAV and the Semantic Web

The Semantic Web [27] is a related approach and has been described briefly in section 2.6. The RDF data model and language [154] is a mechanism for the representation of facts as semantic triples, and as such it is eminently suitable for use as a linkbase layer for Ternary Link adaptation. In this case the "Ternary Semantic Linkbase" may be replaced by an "RDF repository", and then Ternary Link adaptation could be extended to make use of the wide range of semantic resources which are under construction by the Semantic Web community. However, RDF is only one of the possible approaches in establishing such semantic triples and the TRM-NAV provides more possible configurations of those semantic triples (such as dynamic and bi-directional triples) which are not possible in RDF.

When comparing the Semantic Web with the TRM-NAV, it is noticeable that the Semantic Web expresses facts, but not necessarily through the links. The fact that RDF uses triples does not immediately show how the system does in term of linking. There are a few approaches to provide navigational functionalities to the user on top of the Semantic Web repositories, like the Semantic Web Browsers [69], the Semantic Web User Interaction [66, 165] and integrating the Semantic Web with COHSE (section 2.4.5) in [89]. TRM-NAV may be used in modelling the navigational functionalities of such systems, but not the Semantic Web as a whole.

An important noticeable point is that the ultimate goal of the Semantic Web is to provide machine-readable media, while the Ternary Links are intended to be traversable by humans. While many approaches try to convert human-readable facts into machine-readable data in RDF (like [89]), using RDF for knowledge-oriented hyperlinks implies re-converting such data to human-readable fashion (like in [51]), which can consequently cause conversion overloads or a possible accumulation of conversion errors.

This has been partly shown in the example of section 5.5. That example shows a main difference between the goal of the TRM and the philosophy of the Semantic Web: the TRM-NAV uses navigation to provide some meanings to the users, while the links of the Semantic Web is designed to be used by machines.

There are other fundamental issues in comparing RDF with TRM-NAV which will be studied in the following sub-sections.

5.7.1 The Challenge of RDF's Self-descriptivism

Recalling from section 2.6.1, the most important primitive concepts of RDF elements/RDFS are resource, class and property. A reason of this importance is that RDF elements and RDFS are intended to be self-descriptive without need to any external metadata. RDF elements/RDFS are description frameworks, which consequently wish to describe themselves in the same way that they describe anything else [14]. This fact is reducible to the description of RDF's three primitives by themselves. Another interesting reason for focusing on them is for mapping them to the TRM, while everything in the TRM is a node (say, a resource) of the same type.

According to RDF specifications[1];

1- Everything is an instance of *class:Resource* (the class of everything).

2- Every class is an instance of *class:Class* (the class of all classes)

3- *class:Class* is a subclass of *class:Resources.* (because instances of *class:Class* are 'things' so they are instances of *class:Resource*)

4- *class:Resource* and *class:Class* are instances of *class:Class*, since they are classes.

So, *class:Resource* is an instance of *class:Class* and *class:Class* is a subclass of *class:Resource.* Also *class:Class* is an instance of itself (or *class:Class* is of type *class:Class*). It seems that these two statements are the most simplified self-descriptive loop of RDF. However, the meaning of "is-an-instance-of" (or equally, "is-of-type") and "is-subclass-of" are not described yet.

To clarify those terms, RDF specification says:

1- A property is a resource that relates a resource to another resource.

2- *class:Property* is a class of all properties. It is a subclass of *class:Resource.*

3- *type* is a special property (or equally; a special instance of *class:Property*) which relates a resource to a class.

[1] Extracted from these web addresses of w3.org website: http://www.w3.org/1999/02/22-rdf-syntax-ns, http://www.w3.org/2000/01/rdf-schema and http://www.w3.org/2002/07/owl

4- *subClassOf* is another special Property which relates a class to another class.

So, another self-descriptive loop is the fact that: *type* is of "type" *Property*.

The mentioned confusing loops are not necessarily disadvantages of RDF. The fact is that in a description framework, there must be some basic concepts that are either described by external metadata, or be left undescribed, or described by themselves. RDF has selected the third solution.

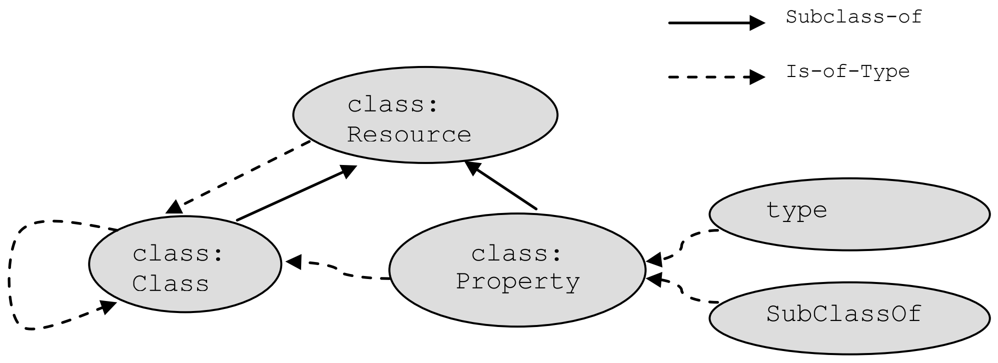


Figure 5-8: Class hierarchy and instances relationship in RDF [14]

Figure 5-8 shows the resulted graph. The main circular facts in the above graph are that 1) the meaning of arrows are defined by some of the circles; and 2) the arrows are making loops between *class:Resource* and *class:Class*.

As will be shown, the TRM avoids the complexity of self-description by 1) keeping its architecture simple and 2) relying on the external descriptions for its primitives.

5.7.2 Basic Limitations of RDF

At the first look, the properties (instances of *class:Property*) are for being used as predicates, and for many of the properties it is almost pointless to be used as object or subject of other statements. For example "is-Taught-by" is a property but there is no point to say something about this term, like "is-Taught-by is …", but a property can have other properties, like *domain* and *range* (in this example, the domain is *class:Course* and the range is *class:Lecturer*) and this means that a statement about a property has been built in which a property is used as a subject (in the example, a statement is "is-Taught-by has domain of Course"). For many other terms, it is reasonable to make real-life statements about a property. For example, the term "phone" can be used either as a predicate to say "The phone of Tim's office is 14231" or as object to say "Tim likes his phone".

This shows that there are also many cases when a single resource is used as a predicate, object, or subject in different statements. But in RDF, although not restricted formally, properties are specialized to be predicate: "Properties are special kind of resources; they describe relations between resources" [14]. That is probably the main reason for the RDF community to have different class definitions for the *class:Property* and *class:Class*. In this way, multiple usage of a single resource as predicate, subject or object, means multiple-inheritance from *class:Property* and some other classes, which seems pointless in RDF context (although valid).

If the example of "phone" is studied again, it becomes clear that in fact, "his-phone-number-is" is the main term of the predicate in the first statement, and "phone" is the object of the second one. Also one can build another statement using "is-phone-number-of" as predicate. This fact was a motivation point for introducing the TRM concept of bi-directionality, in which a single node can have different look, depends on being predicate, object or subject. This is while in RDF, "phone", "his-phone-number-is" and "is-phone-number-of" are three different resources, because RDF resources have a single look. Moreover, there is no RDF-layer mechanism to relate these three resources to each other.

A strict RDF rule says that in such triples, the predicate must be a property [14], which means that one of the three resources in each triple must be an instance of *class:Property*. Since *class:Property* is a subclass of *class:Resource*, there will be members of *class:Resource* who are not qualified to participate in triples as the predicate. Since *class:Resource* has other subclasses like *class:Property*, *class:Class* and *class:Literal*, those unqualified resources can be a class, an instance of a class or a literal value.

class:Property is not documented in RDF specification as disjoint with any other class. This means that to overcome the mentioned limitation, any other resource must be re-instantiated as a property when it necessarily wishes to be a predicate of a statement. However, this implies the excessive complexity of multiple-inheritance in the OO model of information. Also there may be some restrictions on re-defining class membership for existing objects, at least for certain classes of users.

A main difference between *class:Class* and *class:Property* is that instances of *class:Class* are themselves class, meaning that they can have instances, while instances of *class:Property* cannot necessarily be instantiated. A property, by definition, is "A specified aspect, characteristic, attribute or relation used to describe a resource" [105]. A property is sufficient to be a predicate of a statement. For classes, one usually makes statements about instances of some class, not about classes themselves. For example, "Lecturer" and "Course" are classes and "is-Taught-by" is a property. It is not needed to say that "Course is taught by Lecturer" as an RDF statement (because this concept has been already expressed in the domain and range of the is-Taught-by property in RDFS); rather it is much more sensible to say that "Tim is a lecturer" and "Java is a course" and that "Java is taught by Tim" as RDF statements.

5.7.3 OWL vs. TRM in Bi-directionality

Among OWL elements, *inverseOf* is similar to the TRM's bi-directionality. In this mechanism, a property is described to be the inverse of another property. In the TRM also, bi-directionality has been defined and it uses the internal data structure of each node to describe its bi-directional reuse potential and behaviour. Here are the differences:

1- The inverse-property mechanism is working on the ontology layer and not on data model layer; while the TRM-NAV's bi-directionality mechanism is built in the data model layer. In RDF-OWL model, two resources that have been already defined in a lower layer can be defined to have a reverse-of property against each other in an upper layer.

2- In the Semantic Web, two different resources must be defined so that they can be participants of inverse-property; while in the TRM-NAV a single node contains the required information. A node has two different looks when participating in a relation in two directions, as well as a third look when participating as a source or a destination.

3- In the Semantic Web, only a special class of resources (properties) can have inverses, while in the TRM-NAV each node can have bidirectional behaviour. Having an inverse in the TRM-NAV doesn't depend on the allocated role of a node in statements, while in the Semantic Web, it does.

For example, the following OWL lines can express the "reverse-of" relationship between two resources ("teaches" and "is-taught-by") that have been already defined in RDF:

```
<owl:ObjectProperty rdf:ID="teaches">
 <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

In TRM-XML instead, a node defines two associative meanings together when defining a single node of "teaching", as follows:

```
<node>
   <id>thisNodeID</id>
   < desc>Teaching</desc>
   <da>teaches</da>
   <ra>is Taught by</ra>
<node>
```

5.7.4 Interchangeablity between RDF-XML and the TRM-XML

Both the TRM-DB (chapter 4) and RDF can be written in XML. Two main questions can be answered here which can end up with an important conclusion: If some information has been already described by RDF-XML, can it be described by the TRM-XML? What about the opposite direction?

TRM-DB is able to describe any data described in RDF (including RDF schema). A general statement about joining three elements of "thisSubject", "thisPredicate" and "thisObject" in RDF-XML and the TRM-XML has been shown in Table 5-1. It shows that from RDF to the TRM, no information has been missed through the conversion.

Table 5-1: Comparing similar expressions in RDF/XML and the TRM-XML

RDF/XML	<…defining the predicate element as rdf:Property…> <rdf:description rdf:about="thisSubject"> <thisPredicate>thisObject</thisPredicate> </rdf:description>

TRM-XML	<…defining three nodes for thisSubject, thisPredicate, thisObject…>
	<node><id>thisRelationID</id>
	< src>thisSubjectID>/src>
	<asc>thisPredicateID></asc>
	<dst>thisObjectID</dst>
	</node>

In Table 5-1, there are three main differences or missing points in converting from the TRM to RDF:

1- The statement itself has an identity in the TRM-XML (called *thisRelation*), but this is not necessarily a resource in RDF-XML.

2- In RDF, *thisPredicate* must be defined (or re-defined) previously as *rdf:Property*, otherwise the RDF statement is not valid.

3- The TRM relations contains the reverse meanings (through attributes of *da* and *ra* while defining the node of *thisPredicate*), as well as a link to the abstract meaning of *thisPredicate*, but RDF doesn't.

As a conclusion, the TRM can describe RDF-described data, but RDF cannot describe the TRM-described data *completely*.

5.7.5 Conclusion: RDF vs. the TRM

As a conclusion, Table 5-2 summarizes the similarities and the differences between the TRM and RDF both in theory and in application.

Table 5-2: Summary of comparison between the TRM and RDF

Theory		
Issue	**RDF**	**TRM**
Abstracted Layer (Figure 3-4)	Information Model	Fundamental Model
Association as the third element of a relation	Yes	Yes
Treating associations as nodes	Not always possible, generally not applicable	Yes
Treating relations as nodes	Not always	Yes
Bi-directionality of associations	No	Yes
Theoretic Support for functional and dynamic links	No	Yes
Complexity	15 elements (RDF) +15 elements (RDFS) Class hierarchy & Possible multiple-inheritance Self-descriptivism	1 element +8 attributes No class hierarchy No Self-descriptivism
Application		
Issue	**RDF**	**TRM**
General description framework	Yes, for writing *statements*	Yes, for building *relations*
Readability	For machines	For humans
Applicable for global scale	Claimed, Many arguments	Not claimed
Providing data model for AI/ontology purposes	Claimed, Many arguments	Not claimed
A framework for knowledge-oriented hypertext	No	Yes

Also is has been concluded that the TRM-DB can describe any RDF-described information but not vice-versa, and that RDF is based on a subset of the Static-TRM which is directed to satisfy the Semantic Web goals.

5.8 An Abstract Study on the Functional Links

In this section the concept of the functional links will be described in more details and various implications of their existence in BRM, TRM and possibly RDF will be compared. The main meter in this comparison is the concept of "Turing Completeness" and its recent extensions.

5.8.1 An Introduction to Turing Completeness

Alan Turing in 1950 showed that a modelled machine, called Turing Machine, is a formal representation of Algorithms, i.e. it has all the necessary power for doing computational procedures [176]. In fact any computational model for doing Algorithms are reducible to his machine, and that the Turing Machine is the most general model of computing. Turing Machine is a very abstracted machine that works on a "tape" as its input and output. The tape contains series of "cells" and each cell may store a "symbol" of an "alphabet" associated with the machine (like 0 and 1 in the binary alphabet, etc.). At any time sequence, the machine can push the tape forward or backward by one cell, and the machine can read that cell or write on it. The state and the action which the machine performs at any sequence of time is a function of its previous state and the input.

Later in 1956, the Turing Machine was proposed to be the equivalent machine to represent the highest class of programming languages in the Noam Chomsky's Hierarchy [53], which can process phrase-structure grammars and recursively enumerable statements.

"Turing Completeness" is often used for the ability of a system to posses as much expressive power as a Turing Machine, i.e., every computable procedure by Turing Machine to be computable by that system. That system can be –but not limited to- a computational model or a programming or query language. Because of the known expressive power of Turing Machine, the Turing Completeness is often used to analyze the power of an arbitrary system [94].

5.8.2 Functional Linking in the BRM, the TRM and RDF

The BRM was motivated by but is by no means only useful to hypertext navigation. It is in fact a general classification of representations of binary relations regardless of their application or visualisation. Hypertext systems can obviously benefit from the analysis within the BRM as it can clearly delineate the limitations and advantages of some implementations and show how those limitations and advantages arise.

Moreau and Hall [123] discussed the power of various hypertext systems in terms of the Chomsky Hierarchy ([53]), and concluded that implementing hypertext linking in the BRM framework is Turing-complete, as long as it is implemented with the "pE" representation of relations, mentioning that "linking is as powerful as computing". Their conclusion was that "simple links, generic links and some adaptive links all give hypertext systems the power of finite state automata. The history mechanism confers to them the power of pushdown automata, whereas the general functional links give them Turing completeness".

Interestingly, this most powerful representation of a binary relation itself had roots in the modelling of software and verification. The pE, or predicate-Expression, form of a function was how Mili represented binary relations of pre- and post-process states [118]. Namely, the binary relation was not necessarily expressing a semantic relationship but could just as easily express a process. In this context, the Turing-completeness of the pE representation is perfectly natural; as it expresses programs which themselves by definition express anything that is computable.

So if a relation-managing system uses the pE representation of binary relations as its underlying implementation model, it inherits this Turing-completeness, and can do anything that is computable. This is where much of the real power of the Web arises, despite its very basic linking model, because it has the means by which the pE implementation of binary relations can be achieved, combining both process and hypertext linking. Scripting within and of Web pages allows a programmable Web, and while the Web's initial design was not calculated to have this capacity and may not be optimal, it has been successfully grafted onto the existing Web and has much the appearance of a carefully-designed and flexible open hypertext system [134].

Since pE can represent either process or semantic relationships, or indeed any other requirement for a binary relation, there is scope to substitute it into any situation where alternative representations of a binary relation are in use. The prime example that is targeted in this thesis is RDF's use of enumeration to represent semantic relationships.

One feature missing from the BRM representations but present in RDF is the ability to explicitly represent the names of relation incidences from one or more named relations – the BRM representations are unable to incorporate this naming of the appropriate relation or incidence, except implicitly. RDF's semantic advantage is also compromised by the same constraints as those present in the BRM's enumeration of incidences representation, this latter being RDF's relation representation, setting aside the semantics. These constraints include its inability to represent infinite relations, its unsuitability for dynamic use, and its proneness to error over a changeable set of relation elements (in the hypertext case, the endpoints of links).

The Dynamic-TRM unifies the power of the pE representation with the explicit relation naming/designation capability of RDF. Extending the BRM to the TRM exactly addresses this by providing explicit naming/typing/semantic meaning of each binary relation and hence to all binary relation incidences. As a result, a range of semantic relation representations become available, and in particular a semantic pE representation which not only has the highest level of power but which now also can explicitly designate what relation is intended. Thus any relation or collection of relations represented with RDF can be equally well represented with a semantic pE representation, and not lose any of the information about the name/type/meaning of the relation. In fact, a semantic pE representation can be used to determine the full enumeration of incidences for a given relation which can subsequently be stored as RDF triples.

The application of this outcome in implementations of the enumeration of incidences representation is evident. RDF is equivalent to the enumeration of incidences representation of relations in the TRM, with the explicit naming/typing/meaning of the relation present in every relation incidence. Yet RDF is not "powerful" as such, being a data representation, not a programming mechanism. However the step from data representation to programming mechanism is conceptually very simple and consists merely of implementing a different relation representation. The work of this thesis shows how simple it is to incorporate Turing-completeness, perhaps not into RDF as it is at present, but into a marginally modified RDF. If nothing else, the use of an extended pE relation representation can simplify the creation and management of RDF triples within the current infrastructure.

The pE representation has fewer constraints than those representations incorporating enumeration. However it is not able to guarantee "backwards" linking, i.e. bidirectional linking, but otherwise is able to answer the navigational questions within a greater range of scenarios and contexts than other representations.

Moreau and Hall's work [123] thus indicates a significant difference between different relation representations. Enumeration of relations is essentially a data structure. The pE representation renders relations as processes. These processes can always be reduced to data structures by fully evaluating all possible relation incidences, which may be desirable for pragmatic purposes such as performance [22]. However the converse is not necessarily true, as it is not always possible to fix on the "true" processes giving rise to a set of relation incidences.

The benefits arising from referencing relation elements on a "call by nature" basis rather than "call by name" are observable. Including participants in a relation by name ensures that they remain in the relation, regardless of whether subsequent activity renders them irrelevant, and at the same time it means that other elements that might become eligible for inclusion due to meeting requisite conditions are however not included by default. There is no inherent inclusion or exclusion, based on meeting relevant criteria, so that for those relations which were created with criteria-based selection processes, the actual management of selection occurs outside the relation representation.

By contrast, describing participants in a relation by explicitly prescribing the selection process in the representation greatly simplifies the management of the relation. Eligible participants come and go from the relation as appropriate, and it is possible to extract small sub-relations on criteria-based selection as well, so that the entire relation never need be fully manifested. This is discussed further in section 5.

5.8.3 A Turing Completeness Analysis

Having introduced the semantic component to relations has benefits for many hypertext systems. For example, the functional linking (pE) in [179] which was found to be "as powerful as computing" in [123] maps into the TRM-NAV as a "pfE" representation, so that now there will be "semantic functional linking" with transparent semantics to assist users in link choice. It still however retains its Turing-completeness, its ability to operate over infinite data, its ability to select out subsets of the relation incidences without full computation, and still retains the ability to translate into any of the enumerated representations.

pfE is a general case when an unadvertised source of a link can be filtered by the *p()* function, then the available associations are computed by knowing the source, and finally the available destinations are computed by knowing the source and the nominated associations. By analogy to the BRM and using the same arguments as in [123], pfE can support theoretical satisfaction of Turing completeness. The proof is extensible because the BRM is a special case of the TRM-NAV where the association is fixed. Thus a hypertext system that implements pfE linking is Turing-complete, which it inherits from pE linking.

It is evident that the semantic enumeration of incidence representation from the TRM is equivalent to RDF. RDF stores triples in the <from-element, name/type, to-element> format which exactly corresponds to the TRM enumeration of incidences in the form of $\{(x_1, R, y_1), (x_2, R, y_2), ..., (x_n, R, y_n)\}$. What this means is that it is now possible to analyse the relative strengths and weaknesses of RDF and consider possible alternative relation representations which could achieve the same aims plus more.

Hypertext links are often directly analogous to RDF triples in terms of their creation and maintenance, and in fact often hypertext links are used to represent a semantic relation. It is found here that all the challenges of creating and maintaining relation participants, whether links or triples, are the same, and that the analysis of hypertext linking in [19] is directly applicable to RDF. The relevant issues include:

1- *Creation*: making new links/triples easily without too much human input especially on large data collections;

2- *Soundness* [173]: all links/triples are semantically consistent, no false positives;

3- *Correctness*: maintaining technical accuracy of links/triples in the context of volatility within the underlying data collection. (i.e. error 404 occurrences);

4- *Calculated relations*: being able to define relations without creating them in full allows the calculation of a required ternary link. For example, any measuring system that uses the real numbers cannot be calculated in full, so that relations such as "distance-to" must be discretised in an enumerated representation, but could be more accurately represented with a computation.

5- *Targeted subsets*: selecting a small but relevant subset of relation incidences without computing the entire relation.

A central premise of this study is that RDF may be viewed as one of the representations within the TRM. A consequence of this is that the TRM's pfE representation of semantic relations could potentially be used as either an alternative or a supplement to RDF's triples. At present, the generation and maintenance of triples is external to their use. By considering pfE as a mechanism for representing ternary relations, the processing, i.e., creation and maintenance, of triples becomes inherent.

There are two immediately obvious amendments that could use pfE to enhance RDF, both backwards-compatible with current RDF:

1- *Automation*: the minimalist solution is to build software to automatically create and maintain triples in much the same way as links, with all the same considerations for pre-computation of links. This is not actually doing anything to change the Semantic Web softwares or the way triples are received and interpreted, although it will change the way triples are created and perhaps included in documents.

2- *Turing-complete RDF*: to enable RDF to represent semantic relations in any of the TRM forms, but most importantly to expand from having just simply enumerated triples to having triples plus relations represented with pfE. This has the benefits of enabling dynamic relation management so that triples can be created "on the fly" and in response to the context.

The former option is essentially what occurs in those applications that use RDF as a linkbase. The latter option requires a more radical alteration to RDF management software which now must be able to invoke calculations and receive results. However, it is not so far-fetched when the Web is considered to exactly do that at present, with servers calling on external applications to perform calculations even in simple situations such as CGI and PHP scripts.

5.8.4 Discussion

It is important that RDF be made Turing-complete. It has become much more than the metadata language that it was originally conceived as, and is now the backbone of a significant amount of Web development. Limitations in the power of RDF as it stands now will limit future Web development.

RDF is essentially at the same stage that hypertext systems, including the Web, were in during the early 1990s, where everything had to be explicitly named in order to be related. Since then, hypertext systems and the Web have progressed beyond this with dynamic computation of scripts which were in effect pE linking. Relations, in hypertext systems at least, have become both process and data, depending on which of the relation representations are used.

Now it is proposed here that it is time to do the same with the Semantic Web relations. There is a previously-existing model of binary relations, the BRM, which analyses link implementations as binary relations and diagnoses their strengths and weaknesses. If that model is extended to incorporate the notion of explicit naming/meaning of relations into the TRM-NAV, then it can be equally well applied to semantic relation management, such as in the Semantic Web or in workflow management. The costs and benefits have been studied within hypertext and are known [20], and all that remains is to map the procedures onto RDF relations to gain the same benefits.

It is found that RDF maps exactly into one of the representations in the TRM-NAV. This immediately suggests that the many benefits of alternative relation representations that were found in hypertext management could beneficially be applied to semantic relation management in the Semantic Web.

Considering then the management of semantic relations, there is clear benefit from augmenting the forms of relation representations available to the Semantic Web applications. Conceptually, this augmentation is very simple indeed, and consists of adding another form of relation representation, the pfE representation, to the Semantic Web applications. In fact, it need not even be directly added to existing applications, accessed by many, but rather could follow the open hypertext systems model of publishing sets of ready-made relations which anyone can use but are managed by the owner [45].

But at its most promising, the many benefits of the pfE representation include the ultimate in flexibility – the ability to define relations as processes, not just as data. By enabling call-by-nature relation management to supplement RDF's call-by-name relation management, a criteria-based relation management is gained here, with many useful features.

Users frequently want to classify data according to the "nature" of data and thus allocate them to relations on that basis. Algorithms that specify the selection of elements for participation in a relation are obvious improvements over the manual creation of relations in such cases. Such improvements include the automation of relation incidence creation, with the attendant benefits of relation *soundness* (no false positives) and relation *completeness* (no false negatives), as well as reducing human intervention.

As a result of this, every time new data is exposed to a relation's selection process, it can be assessed for its eligibility automatically. Thus a user can define their own relations, based on selection criteria relevant to their own purposes, and apply them to whatever data they encounter. The example is section 5.3.1 could have different pfE relations for each user, so that one user could have links from "Malaysia" but another could have links emanating from family member names instead. Also, users often want to subsequently *search* for relations or incidences from relations based on their "nature". When the selection is already a part of the process to make relation incidences, the process is simplified.

More importantly, it is possible to perform this selection as a dynamic process. This is important not only because of avoiding unnecessary computation, but also because it gives relation *correctness*. A relation or incidences calculated upon request will be correct at the time of the request, more likely so than something that was correct at some earlier calculation date [20].

Another benefit of the dynamic computation is that it bypasses the need to create or manifest all of the incidences in the relation, yet the required instances are still available. Dynamic computation using pfE enables this, with the predicate part of the computation selecting out the relevant subset of relation incidence sources according to the appropriate criteria. This creation of targeted subsets of relations can be useful whenever there is no need to create all incidences in a relation just to access a subset of them. In particular this is useful in a volatile data environment, where relation correctness potentially becomes an issue as relation elements themselves change in terms of the selection criteria.

5.8.5 Section Conclusion

The BRM showed that binary relations can most usefully be represented with Mili's pE representation, and Moreau and Hall concluded that hypertext systems which use the pE representation as the basis for their linking (relation) activity are "as powerful as computing", and thus are Turing-complete. Mili actually used the pE representation as a means of expressing computer programs. Enumerated binary relation pairs however are not as powerful, being merely a list of participating entities.

However the BRM, and hence the pE representation, does not explicitly represent the name or meaning of the binary relation, this being only implicit in the definition of the relation. In contrast, RDF explicitly names the relation to which each pair of entities belongs.

RDF represents semantic relationships with triples, explicitly naming two participating pairs of individual entities and the relationship which binds them. The enumeration of related pairs within relations is however only one method amongst many for representing relations, and not always the most efficient or viable.

The value of the TRM in the functional linking is firstly that RDF's triples can be considered within a framework and compared to alternatives, such as the TRM version of pE and the designated pfE. A system whose relations are represented with pfE is likewise going to be Turing-complete. Secondly, if RDF can be equipped with a pfE representation of relations, this would give far greater power and flexibility within the Semantic Web applications. The management of RDF triples is reducible to the management of ternary relations and manifests many of the same issues as are found with the management of hypertext links.

The Semantic Web is in some respects in the same position as hypertext systems were when only point-to-point linking was available. The problem at hand is essentially the same – how to manage the creation and maintenance of large numbers of relations and relation incidences over a changing, many-owner collection of information. There are differences in the purpose and usage of those relations but this does not affect the technical challenges of managing them. Casting relations into a model helps us to forget the superficial differences in the use of relations, and to apply lessons learned from one relation-management situation into another.

What was found from the management of hypertext links is that a conceptually simple change in the way relations are represented can significantly alter the capability of hypertext systems, in some cases translating what is in some representations nothing more than a data structure into what becomes, in another representation, a process, and thereby gaining all the advantages of programmability without sacrificing the ability to render those processes back into data structures as needed. This can also be done with semantic relations.

5.9 Summary

In this chapter, TRM-NAV has been introduced as an extension to the BRM in modelling hypertext navigation. Also Ternary Links as a visual representation of this model has been reviewed. The potential of the TRM-NAV in adaptive Hypermedia application and were also discussed. As an extension of BRM, TRM-NAV inherits all of the beneficial characteristics of BRM, particularly providing Turing Completeness for the built hypertext systems. Moreover, the semantic functional links of TRM-NAV provides more knowledge-orientation to the hypertext system than the functional links of the BRM. TRM has also been compared with data model layer of the Semantic Web in this chapter.

C h a p t e r 6 -

TRM-WF: A NEW WORKFLOW DEFINITION MODEL

This chapter introduces a new model and notation for workflow modelling called "TRM-WF" (TRM-WorkFlow). After revieweing the main related works on workflow modelling, the next step is showing that the studied models and notations can be rewritten or reduced to the TRM. Then the TRM-WF will include the TRM notations for workflow definition and the method of storing workflow definitions in storage layers; e.g. in the TRM-DB (chapter 4). As an instant result, the TRM-XML can be considered as a process definition language. It will be shown that this model can provide simplicity, re-usability, bi-directionality and dynamic characteristics to the modelled workflow.

6.1 Related Works on Workflow Modelling

In this section, the specifications of some known workflow models will be studied. These related works include: Petri nets, Wf-nets, UML, BPMN, XPDL and YAWL. They are models, languages and/or notations that are widely known and used in the workflow community and are referenced by WFMC documentations [195]. In addition, the Workflow Patterns will be introduced as a standard ruler for workflow modelling.

6.1.1 Petri Nets

The term of "Petri net" has been coined by Carl Adam Petri in his PhD thesis written in 1962 [146]. Petri net has both mathematical and graphical definitions and notations and the formalization of Petri nets over the past decades makes it powerful for modelling and analyzing processes. As will be shown, workflows are a subclass of Petri nets and this can provide a mathematical support for analyzing workflows [146]. Unlike workflows, a Petri net has a formal definition, so there is not any formal proof to the fact that Petri-nets can explain all workflows in the world. Instead, using Petri-nets for workflows has been inspired by practical experiences and it has been generally agreed in the workflow literature that Petri net formalism is useful in the context of workflow management. In fact, only a part of the workflow management, which is "control-flow", can be modelled by Petri nets [1, 3]

A Petri net in its static graphical definition is a "directed bipartite graph". A bipartite graph is a graph with two separated types of nodes while no edge can connect two nodes of the same type [49]. For Petri nets, these two types of nodes are called "Places" and "Transitions" and the directed connectors are called "Arcs". The graphical notations of them are circles for places and rectangles for transitions, so connections between two places or between two transitions are not allowed. Arcs can exclusively connect places to transitions or transitions to places. Petri net is a directed graph because the arcs have explicit directions. Petri net graphical notations are listed in Table 6-1.

Table 6-1: Petri-net notations [146]

Element	Description	Notation
Place	Placeholders for tokens, where they wait for firing by transitions.	○
Transition	An action that can transfer tokens from input place(s) to output place(s)	▭ Or ▬
Arc	A connection between places and transitions or transition to places.	⟶
Tokens	Each instance or case to be located in places	●

In terms of dynamic characteristics, places of a Petri net may contain "Tokens" and these tokens can move from one place to another through transitions. Tokens will be mapped into workflow cases (or work items) later in this chapter. The "State" of a Petri net is known by the placement of tokens inside the places of a Petri net in a certain time slot. The process of passing a token between places through transitions is called "Firing" of that transition. A transition can fire if it is "Enabled" which means that there is at least one token at each of its input places. By firing, one token is removed from each input place and one token is added to each output place. A sample firing process has been shown in Figure 6-1.

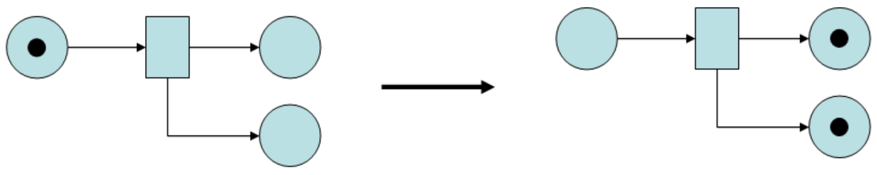


Figure 6-1: A sample Petri-net firing

Formally [33, 34, 36], a Petri net is a tuple of (P,T,F,M(t),W,K) where:

- P is a set of places
- T is a set of Transitions, while $P \cap T = \varnothing$
- F is a set of arcs, while $F \subseteq (P \times T) \cup (T \times P)$
- M(t) is the state of Petri net at time t, a dynamic mapping from P to IN (IN is the set of natural positive numbers)
- W is a set of arc weights, a static mapping from F to IN.
- And K is a set of place capacity restrictions, and a static mapping from P to IN.

The triple of (P, T, F) can completely express the graph of a Petri net in a static view. M(t) represent the dynamic token distribution of the Petri net over the periods of time. W and K are optional sets that add extra information over a Petri net basic definition to make it functional in real-world conditions and they will not be covered anymore in this thesis. More information can be found in [3, 67].

6.1.2 Wf-Nets

Although Petri net provides enough basis to explain workflows, but there is no guarantee that every Petri net is equivalent to a workflow. For instance, workflows by nature must have starting and ending places, and it cannot be an endless loop. Thus, workflow must have a place with no entering arcs (a starting place) and another place with no outgoing arcs (ending place). Also the routes from the starting node to the ending node must cover all other nodes and there shouldn't be any isolated node in a workflow, i.e. nodes that cannot be passed ever. These characteristics are not covered in the formal Petri net definition. Instead, a Petri net that can define a workflow is called a Wf-net by Aalst in his different works [33, 34, 37-39] and has been used widely by workflow community (WFMC) [195].

Wf-net by definition [3], is a Petri net with the following conditions:

i. There is only one source place that has no entering arc (source node)

ii. There is only one sink place that has no outgoing arc (sink node)

iii. Every other node is on a path from the source node to the sink node.

The above definition of Wf-nets can be rewritten mathematically as: A Petri net defined by a triple of (P, T, F) is a Wf-net where:

$$\exists i \in P \therefore \forall x \in T, (x,i) \notin F$$

$$\exists o \in P \therefore \forall x \in T, (o,x) \notin F$$

$$\forall x \in P \cup T, \exists x_1, x_2, ..., x_n \in P \cup T \therefore (i, x_1), (x_1, x_2), ..., (x_i, x), (x, x_j), ..., (x_{n-1}, x_n), (x_n, o) \in F$$

In the above mathematical definition, the first two conditions have not explicitly limited the number of source nodes and sink nodes to be only one, however, the third condition can do this job. This is because if there are two source nodes, then one of them must be in a path from another to the sink node. Being in a path means having both entering and outgoing arcs from and to this source node, which is impossible. The same reasoning can be applied on the case of the sink node.

Moreover, being a Wf-net doesn't mean being a completely valid and working workflow. There are other characteristics that make a Wf-net valid. For example, the above definitions cannot test a given workflow against dynamic problems like endless loops or dead-ends. Thanks to the formal definitions of Petri nets and Wf-nets, these dynamic characteristics can be analyzed mathematically on a given workflow. The validity of workflows and properties like soundness, free-choice, well-structured and boundedness is not covered in this thesis and more information on them can be found in [3].

6.1.3 UML Activity Diagram

The Unified Modelling Language (UML) includes a set of standardized modelling languages and notations in the field of software engineering, system engineering and business process modelling [103]. UML is a developing standard initiated by Object Management Group (OMG) [191] and the last version at the time of writing this thesis is UML 2.1.1 [139]. One of the graphical notations in UML is the "Activity Diagram" which is known to model any type of "flows". In fact, what is known as "Flow Chart" in many different fields of science and engineering has been standardized as UML Activity Diagram. Because workflows are widely expressed graphically as flow charts, UML Activity Diagram can be used in workflow modelling [70, 148].

In defining workflow definition in UML Activity Diagram [3, 70], the diagram is divided into some "Swimlanes" (shown as vertical lanes), each representing a "role" in the workflow. In each swimlane, there are some "Activities" (shown as rounded rectangles) to represent the workflow tasks to do, some "Branches" (shown as diamonds) to represent the decisions, and "Synchronizations" (shown as thick horizontal bars) that can be either a "Fork" (splitting point) or "Join" (merging point). Arrows can connect any two thing either being activity, decision or synchronization from and to any swimlane. Starting and ending points are also shown as circles. In addition to control flow, the Activity Diagram can also denote the object-flow, which is not essential in a workflow definition. A sample workflow has been shown in UML Activity Diagram in Figure 6-2.

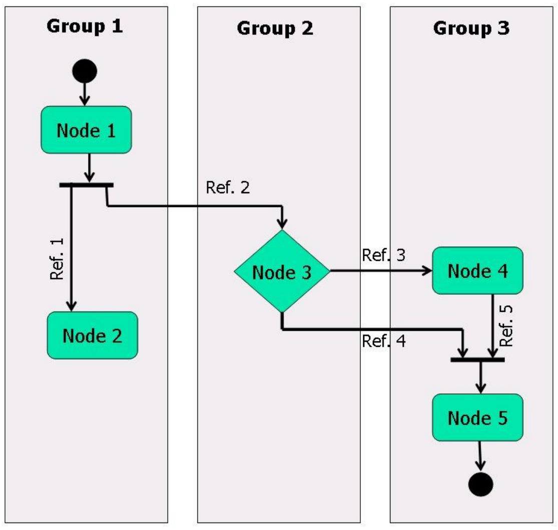


Figure 6-2: A sample UML Activity Diagram used to define a workflow

There is an important relation between UML Activity Diagram and Petri nets: Every workflow described in the UML Activity Diagram has a Petri net equivalent [3]. The mathematical proof of this fact is out of the scope of this thesis but the result will be used later to integrate different modelling approaches.

6.1.4 BPMN

BPMN (Business Process Modelling Notation) [190, 197] is another graphical notation to define workflows. The development of BPMN has been started by BPMI (Business Process Modelling Initiative) [189] in 2004 and then continued by OMG (Object Management Group) [191] after two organizations merged in 2005 [189]. BPMN has been widely supported in the documentations released by WFMC (WorkFlow Management Coalition) [195]. The final adoption of BMPN 1.0 specification (2006) can be found in [138].

Unlike UML and Petri nets, BPMN is specially directed to support workflow technology by providing standard diagrams. BPMN diagrams look like common flow charts but with added special notations to support parallel processing, joins, etc. According to BPMN specifications [138, 197], the basic categories of elements in BPMN are summarized in Table 6-2. A sample workflow represented in BPMN has been shown in Figure 6-3.

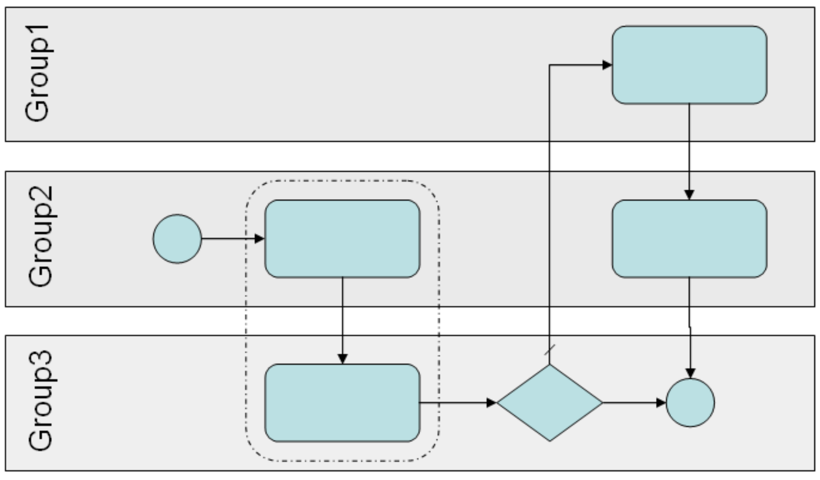


Figure 6-3: A sample workflow represented in BPMN

Table 6-2: BPMN notations [138, 197]

Category	Element	Note	Notation
Flow Objects	Event	Things that happens in the real world and affect the process of workflow, including start and end of a workflow.	○
	Activity	A task that must be done in each stage of a workflow.	▢
	Gateway	Used to control decisions, merges and splits of the flow. Varieties include "Exclusive", "Inclusive", "Complex" and "Parallel".	◇
Connecting Objects	Sequence Flow	Shows the sequence order of Flow objects. A hatch mark on the arrow specifies the default route when different arrows come out from a gateway.	⟶
	Message Flow	Used to show the flow of messages between flow objects.	- - - -▶
	Association	Used for associating any text, data or notes to a flow object.	··········▶
Swimlanes	Pool	Denotes the groups of participants of the workflow	Horizontal bars
	Lane	Denotes each participant	divided lines within pools
Artifacts	Data Object	Shows data forms/storages to be filled/stored in each activity.	🗋
	Group	A subset of workflow that can be grouped as a macro activity.	⬚
	Annotation	A placeholder for any text, data or notes to be associated to a flow object.	⋱[

For each element in Table 6-2, there are different varieties of graphical notation to show different sub-types. For example, a gateway can be used for decision, parallel split, merge, etc. which are not covered here for simplicity. The value of modelling with BPMN is its coverage over many different notations in business process as a new standard (including UML Activity Diagram introduced in 6.1.3). A list of those notations can be found in [197].

6.1.4.1 XPDL

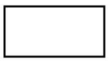
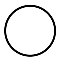
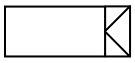
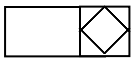
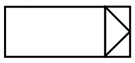
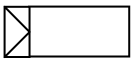
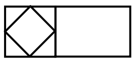
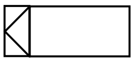
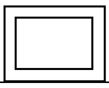
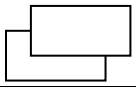
XPDL (XML Process Definition Language) is an XML-based language to define workflows. It is a standards language of WFMC [195] and the specifications of the latest version at the time of writing this thesis (XPDL 2.0) has been described in [196]. XPDL does not provide a graphical notation and it has been developed so that each BPMN diagram can have one XPDL equivalent XML listing and vice-versa. The XML elements and attributes defined in XPDL are designed to represent graphical elements of BPMN and that is how XPDL can "serialize" a BPMN graph. However, XPDL shows no global acceptance as a formal basis of workflow specification [2].

6.1.4.2 YAWL

During the past decades many different languages with different graphical and textual notations for workflow definitions have been developed (a review can be found in [137]) but the reason behind selecting YAWL here is in its theoretical background and the main designer people. Will Van der Aalst is a known name in workflow community which has done much theoretical work on workflow management systems (like [33, 34, 37-39, 50, 52-59]). One of his main works is introducing the concept of "Workflow Patterns". Workflow Patterns are one of the main meters used to analyze the power of a workflow management system: The more patterns to be supported, more powerful the system is [7, 99]. The Workflow Patterns will be reviewed in more details in chapter 6. Although these patterns can be represented by notations like UML Activity Diagram and BMPN [198], but Van der Aalst and his team has specially developed YAWL (stands for Yet Another Workflow Language) as the first language to comprehensively support all of those patterns in its clearest way [50, 58, 59, 62]. YAWL is a set of graphical notations to define workflows and its full description can be found in [4, 5]. An XML equivalent of YAWL has also been developed but is not a core of documented YAWL and can be found in YAWL website [192]. This website also includes how to represent all workflow patterns in YAWL notation.

Table 6-3 contains the main YAWL notations.

Table 6-3: YAWL notations [192].

Element	Note	Notation
Atomic task	A single task to be performed by a human or an external application.	
Condition	A way to represent state for the process (not to be confused with decision), including start and end of the process.	
Split task	Used when the process is divided to more than one route. AND-split for sending the token to all of outputs, Or-split for sending to any number of them, and XOR-split to send it to only one of the outputs. OR and XOR splits are equivalent to decisions.	 (AND)  (OR)  (XOR)
Join task	Used when more than one routes are going to merge. AND-join is activated when all the input tasks has a token, OR-join is activated when at least one of the input tasks has a token and XOR-join is activated when only one of the input tasks has a token.	 (AND)  (OR)  (XOR)
Composite task	A container for another YAWL process - with its own set of YAWL elements.	
Multiple instances of a task	There exists the ability to express that you wish to run multiple instances of a task concurrently	

6.2 Workflow Ternary Transitions

Recalling the ternary approach to the workflow modelling concepts (section 2.7.2) since the purpose and decision factors motivating a transition between two states in a workflow was not explicitly captured in a purely binary relation, workflow modelling needs to explicitly include a third entity associated with every related pair of entities (source and destination), namely the "association" which describes the decision factors behind the use or selection of the transition. This third entity might seem functionally unnecessary, and indeed it could be argued that the mechanics of a binary relation do not require any form of "typing" or semantic justification in order to function correctly.

However, as found in hypertext systems, it is not just the computing agent involved in the manifestation of links or processes, there is very often a second agent involved, namely the human/users who select what the judge the most appropriate link or process from an otherwise indiscriminated set of possibilities. That is, the third entity in the relation is a discriminator that represents the purpose or other semantic characteristics of the association between the other two entities.

The ability to transform data between representations means that existing models can be represented and their functionality can potentially be upgraded. Relations or processes can be represented either as named and explained pairings of states, or as computational descriptions of participating states, with the computation describing the characteristics an entity would possess to allow it to belong to the transition between states. The ability to convert from one representation is a significant alteration, as it means the entities participating can be treated both as data and as processes.

Applying the TRM to process modelling is thus backwards-compatible as well as forward-looking. The workflow model which is based purely on the introduced TRM, called the TRM-WF.

6.3 The TRM-WF Overview

A workflow system contains not only information about definition of a process, but also provides user's non-linear navigation between its nodes. The navigation between nodes of workflow is the ability of a system to guide the user to go from a node to another depending on his/her decision on provided choices. Thus each navigation operation consists of three parameters: source, *decision* and destination. This decision-based navigation has three navigation elements as in the TRM, making it suitable to be built directly on top of the Dynamic TRM.

The TRM-WF has many similarities to the TRM-NAV explained in the previous chapter because workflow navigation has almost the same principles as knowledge-oriented hypertext navigation, to hyperlink navigation, and the semantic functional links (pfE's) in workflow context includes the following node selections:

1- *Source Selection*: The user selects the source node of the navigation, which is the node where one of the current work cases must wait to be processed. This can be equal to selecting the desired item in the user's to-do list.

2- *Decision Selection* (equal to the TRM's association): The user selects which type of processing he/she wants to do on the current work case. Usually the decisions are source-dependant, i.e. the user selects its decision from a list of available choices, which are either predefined or computable to be available on the source node (computed associations). However, there are possible source-independent decisions, like suspension, cancellation or emergency jumps (enumerated associations).

3- *Destination Selection:* The user and/or system determine the destination node(s). The destination can be more than one node, like distribution of a task among the users. If the user selects a source-dependant decision the system determines the destinations (computed link destinations), otherwise the user selects the destination explicitly (enumerated link destinations).

It is noticeable that the word *decision* is used here for multiple purposes. First, it can be a normal decision taken by a user in a process; secondly it can be a description of task completeness in a single output node; and thirdly it can be a real world condition in a solution process. Generally speaking, decisions (associations of the TRM) and destinations are both computed functions and the ternary relation model of workflows can be expressed as $R=\{(x,d,y) \mid \text{some } x \in S\ ,\ d=f(x)\ ;\ y=g(x,d)\}$; where x is source, d is decision and y is destination. Comparing this formulation to the Dynamic-TRM (section 3.1.2), it can be seen that the Dynamic-TRM can cover a workflow definition.

The definition of a workflow system can also be realized by a hypertext system having Ternary Linkbase(s). The nodes are processes that can be done on a work case and they have been abstracted from the links, which are how users can pass the work case among themselves. It is also possible to define multiple linkbases over a fixed set of nodes. In this case it is actually multiple workflows definable in an organization (for business applications) or multiple solutions definable for a specific problem (for solution processes).

Practically, each task is represented in a node (like a Web page) and some other nodes are dedicated to the allowed decisions. Completing a decision task in this workflow system is carried out by following a link between source (initial step) and destination (final step) through that decision node. This has been illustrated in the TRM notation in Figure 6-4.

As an instant result, a dual meaning for each link provides some benefits. The reverse link

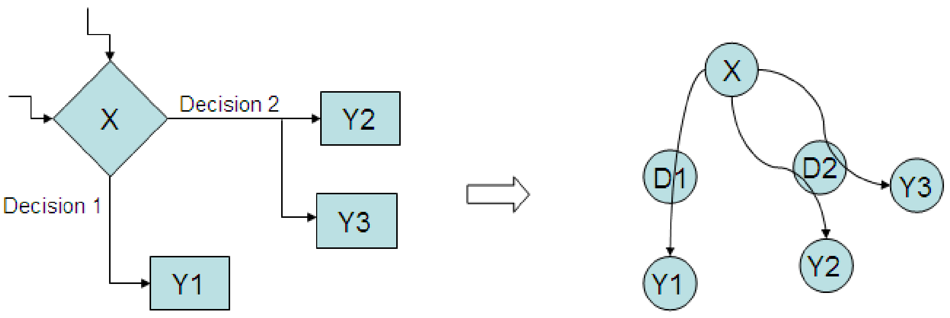


Figure 6-4: A sample decision task and its TRM equivalent

meaning in workflow is actually the reverse description of a decision -if applicable- which particularly means *giving up* a task or *withdrawal* of a case.

Recalling the layered approach proposed in section 3.3, the TRM-WF is located on the top layer of the information model. It means that the TRM-WF is based on the TRM and can be written in alternative storage methods and languages including the TRM-XML. It also means that the TRM-WF is not an application by itself but it can provide the basis for such applications' design.

6.4 The TRM-WF Coverage over the Other Models

In section 6.1 a number of known related works on workflow modelling has been reviewed. In this section the ability of the TRM to cover these related works will be studied. The study is based on comparing the TRM graphical notation with other workflow notations and the coverage of the TRM notation on the Workflow Patterns. Although it will be shown that the TRM can represent many other notations, it is important to notice that the TRM notation is not always recommended to be used instead of any other notation. The provided notation is only to justify the coverage of the TRM as a general framework over other related works.

6.4.1 *The TRM-WF and Petri Nets*

The TRM notation can represent a Petri net. Since Petri net has two static and dynamic characteristics, the proof of the claim must be done in two sections. For static characteristics, the TRM notation must be able to represent a static Petri net graph and for dynamic characteristics, the TRM must have some solutions to represent the state of a Petri net.

6.4.1.1 *Static representation of Petri net*

As studied in section 6.1.1, the graph of a Petri net consists of a number of places and transitions while the arcs connect places to transitions or transitions to places. The fact that two places can be connected through transitions is similar to the fact that two nodes in the TRM can be connected via another node. As the first step to convert a Petri net to a TRM graphical notation, all places and transitions of a Petri net can be replaced with the TRM nodes. Then each two consequent arcs between two places (a place-transition arc plus a transition-place arc) can be replaced with a TRM link that connects three nodes (two places plus one intermediate transition). This can be illustrated in Figure 6-5.

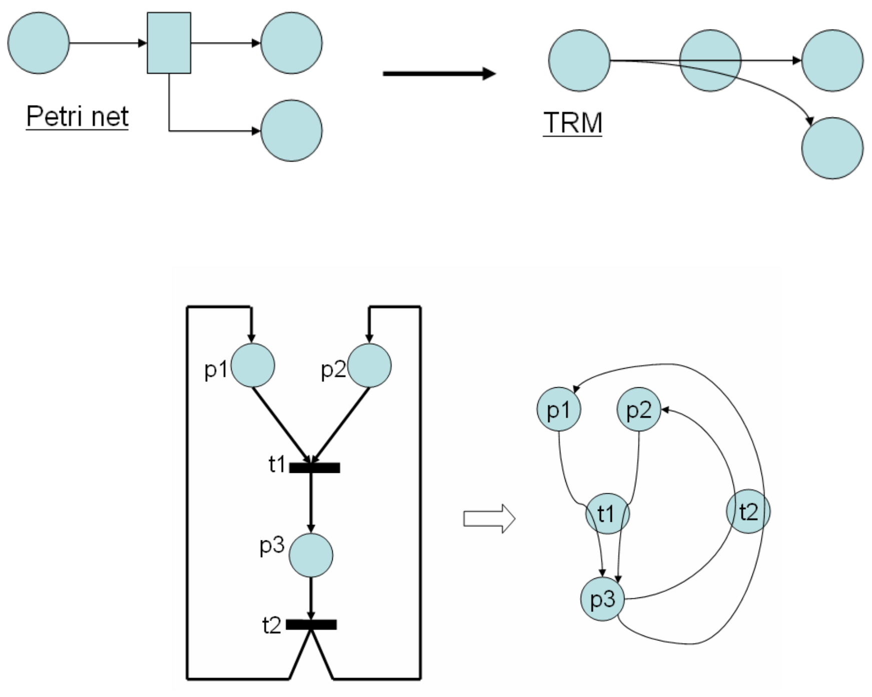


Figure 6-5: Petri-net to TRM conversion sample

Recalling section 6.1.1, if a Petri net graph is a tuple of (P,T,F) then a TRM representation of that Petri net graph is simply a set of N_1 where:

$N_1 = P \cup T \cup F'$ and $F' = \{\textit{some triples } (x,r,y) \mid (x,r) \in F \textit{ and } (r,y) \in F\}$

The above definition fits into the Static-TRM definition of section 3.1.1. An illustration of this arrangement has been shown in Figure 6-6.

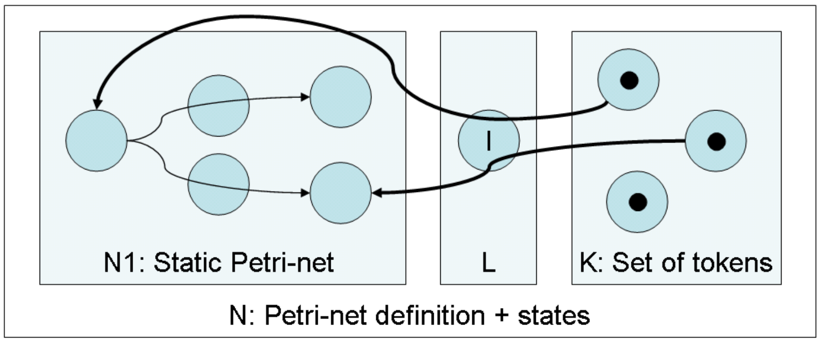


Figure 6-6: Illustration of the Petr net including the token states

6.4.1.2 Dynamic representation of Petri net

As stated in section 6.1.1, the state of a Petri net is the number of tokens distributed among places in at a certain time. Also it will be explained in section 6.4.7 the number of tokens in each place is not enough to show the state of a Petri net completely. The state can be considered as the list of tokens in each place at a certain time. This definition can consequently give the number of tokens in each place as required by the first definition.

Since everything must be defined as some nodes in the TRM, a token is also a node that can be located in a place (another node) using a relation through a special node of "locating". Changing the location of a token in the Petri net is equivalent to changing the destination node of that relation in the TRM.

Formally, the TRM representation of the state of a Petri net is a set of N_2 where:

$N_2(t) = K \cup L \cup R(t)$ and;
K=Set of tokens
L=A node of "Locating" with single member "l"
$R(t) = \{\textit{some TRM triples } (x,l,y) \mid x \in K, y \in P, x \textit{ is located in } P \textit{ at time } t\}$

Finally, the set of $N(t)= N1 \cup N_2(t)$ is a full TRM representation of the Petri net. (P and N_1 has been already defined in section 6.4.1.1)

The above definition is compatible with the Dynamic-TRM definition of section 3.1.2.

6.4.2 The TRM-WF and Wf-Nets

Since Wf-Net (section 6.1.2) is a special case of Petri net and the TRM can represent any Petri net (section 6.4.1), then the TRM can represent any Wf-Net.

6.4.3 The TRM-WF and UML Activity Diagram

It has been mentioned in section 6.1.3 that every workflow described in an UML Activity Diagram has a Petri net equivalent. Using the result of section 6.1.2 , the TRM is able to represent any UML Activity Diagram. Moreover, the conversion from an UML Activity Diagram and the TRM can be done by converting each labelled arc to a node crossed by a connection.

An UML Activity Diagram also supports swimlanes in order to differentiate nodes by their rules in the organization. The TRM representation of this can be done by having a node for each swimlane. Then other nodes can be related to swimlane nodes via another special node called "Belonging". This has been illustrated in Figure 6-7.

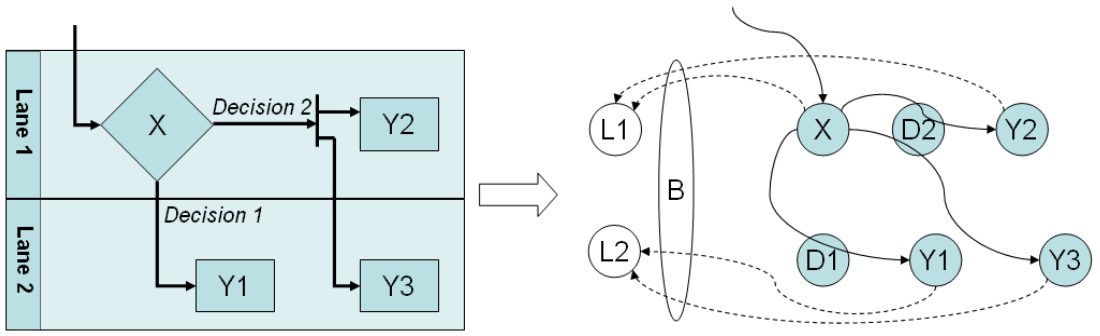


Figure 6-7 A partial example of UML Activity Diagram and its TRM equivalent

6.4.4 The TRM-WF and BPMN

BPMN which is described in section 6.1.4 has similar core elements to the UML Activity Diagram. The main differences are supporting Message Flow, Association, nested swimlane and Artefacts (Data Objects and Groups). The TRM representation of an UML Activity Diagram (section 6.4.3) is flexible enough to support these differences. For message flow, different set of the TRM relations can be used. For Association, the TRM has built-in association support and any node (including text, data, etc.) can be associated to any other node. A Nested swimlane can be covered by having two sets of "Belonging" relations: relations between Nodes and Lanes, and relations from Lanes to Pools. Also Data Objects can be some TRM node, and they can be related to other nodes through special nodes called "Fill" or "Store". Finally, a subset of workflow that must be grouped in BMPN can be adopted in the TRM by relating all nodes inside that subset to a node representing that group. This relation can go via another special node called "Grouping".

6.4.5 The TRM-WF and XPDL

The TRM can represent an XPDL (section 6.1.4.1) by two approaches. First, as described in section 6.1.4.1, XPDL is a serialization of BPMN graph, so if the TRM can represent a BMPN graph, then it has already represented XPDL. Secondly (as studied in section 4.1), any XML listing (including XPDL) can be converted to the TRM.

6.4.6 The TRM-WF and YAWL

As described in section 6.1.4.2, YAWL is specifically designed to support the Workflow Patterns. Because of this, instead of studying how the TRM can represent YAWL, the coverage of the TRM over the Workflow Patterns will be studied in the next section.

6.4.7 The TRM-WF and the Workflow Patterns

In order to provide a conceptual basis for workflow technology, the Workflow Patterns [38, 39, 57] have been introduced in 1999 by a joint effort initiated between Eindhoven University of Technology (led by Wil van der Aalst) and Queensland University of Technology (led by Arthur ter Hofstede) [201]. The Workflow Patterns are possible situations in real workflows that need to be supported by elements of a workflow model, or the potential capabilities that a workflow management system may have [198]. This can be represented by single elements or notations or some combinations of them. Then the power of a workflow model, language or notation can be measured by its ability to represent those patterns. Such an evaluation has been shown in [201] to asses different products and standards in the workflow community. This can be helpful in measuring the power of the TRM-WF.

The Workflow Patterns have been introduced as 20 patterns in [7]. Recently the developers of the Workflow Patterns have revised and categorized them into 4 different groups: 43 Control Flow Patterns in [160], 40 Data Patterns in [161], 43 Resource Patterns in [162] and a tree of Exception Handling Patterns in [159]. The focus of this research will be on the list of 20 Patterns in [7] which is now the main part of the Control Flow Patterns. The study on the recent revised list of the Workflow Patterns is out of the scope of this research. Those 20 patterns have been counted as 19 patterns in [192], 21 patterns in [198] or 26 patterns in [6], but these differences are not essential and they can be mapped into 20 patterns of [7].

Table 6-4 lists 20 Workflow Patterns with their names and brief description and will be used as reference for later use.

Table 6-4: The Workflow Patterns - Control Flow [198]

Category	#	Pattern Name	Pattern Description
Basic Control-Flow Patterns	1	Sequence	Enabling a task after the completion of a preceding task.
	2	Parallel Split (AND-split)	The divergence of a branch into two or more parallel branches.
	3	Synchronization (AND-join)	The convergence of two or more branches into a single subsequent branch such that the subsequent branch is enabled when all input branches have been enabled.
	4	Exclusive Choice (XOR-split)	The divergence of a branch into two or more branches such that only one of the outgoing branches can be enabled.
	5	Simple Merge (XOR-join)	The convergence of two or more branches into a single branch such that an enablement of an incoming branch is enough to enabling the output.
Advanced Branching and Synchronization Patterns	6	Multi-Choice (OR-split)	The divergence of a branch into two or more branches such that some of the outgoing branches can be enabled.
	7	Structured Synchronizing Merge (OR-join)	The convergence of two or more branches (which diverged earlier in the process at a uniquely identifiable point) into a single branch such that the control is passed to the output when each active input has been enabled.
	8	Multi-Merge	The convergence of two or more branches into a single branch such that enablement of one or more input results enabling the output.
	9	Structured Discriminator (1-out-of-n join)	The convergence of two or more branches into a single branch following an earlier divergence. Enabling the output results cancellation of other tokens produced in the divergence point.

Iteration Pattern	10	Arbitrary Cycles	The ability to represent cycles in a process model that have more than one entry or exit point.
Termination Pattern	11	Implicit Termination	A given process (or sub-process) instance should terminate when there are no remaining work items that are able to be done.
Multiple Instance Patterns	12	Multiple Instances without Synchronization	Within a given process instance, multiple independent instances of a task can be created.
	13	Multiple Instances with a Priori Design-Time Knowledge	Pattern 12 when the required number of instances is known at design time and it is necessary to synchronize the instances at completion before any subsequent tasks can be triggered.
	14	Multiple Instances with a Priori Run-Time Knowledge	Pattern 12 when the required number of instances may depend on a number of runtime factors, including state data, resource availability and inter-process communications, but is known before the task instances must be created. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered.
	15	Multiple Instances without a Priori Design-Time Knowledge	Pattern 14 when the required number of instances is not known until the final instance has completed. At any time, whilst instances are running, it is possible for additional instances to be initiated.
State-Based Patterns	16	Deferred Choice	A point in a process where one of several branches is chosen based on interaction with the operating environment. After the decision, other branches executions are withdrawn.
	17	Interleaved Parallel Routing	A set of tasks has a partial ordering defining the requirements with respect to the order in which they must be executed. Also no two tasks can be executed at the same time
	18	Milestone	A task is only enabled when the case is in a specific state (typically a parallel branch).
Cancellation Patterns	19	Cancel Task	An enabled task is withdrawn prior to it commencing execution.
	20	Cancel Case	A complete process instance is removed and recorded as unsuccessful case.

As an example, Figure 6-7 which contains a part of a flowchart, includes the Workflow Patterns number 2 (AND-split) and number 4 (XOR-split).

The main tool for representing the Workflow Patterns in is an extension to Petri nets called Coloured Petri net (CPN) [7]. CPN uses values (or colours) to differentiate tokens travelling in a Petri net. The transitions fire independently for each valued (or coloured) token based on the values associated to each token. In addition, it is allowed to put some “pre-conditions” for each transition. A pre-condition specifies a logical condition over the values of the tokens that makes a transition active if the relevant token exists in that transition’s input place(s) [3]. In this way, an extended Petri net notation is able to illustrate the Workflow Patterns. It is also interestingly shown that BPMN and UML 2.0 Activity Diagram can represent the Workflow Patterns [198]. Also as mentioned in the previous section, YAWL is intentionally designed to support the Workflow Patterns. Finally the TRM-WF is the next candidate to be measured by the Workflow Patterns.

Each of the 20 patterns has a Petri net equivalent [201] and also one or more UML Activity Diagram equivalents [198]. Immediately it can be concluded that all that patterns have a TRM equivalent as well. For a more sensible approach, the main focus can be on the first 5 patterns which are named Basic Control Flow Patterns. These 5 patterns with their Petri-net graph, their UML/Flowchart graphical notation, their YAWL equivalent [192, 201] and the proposed TRM equivalent graph have been shown in the rest of this section.

6.4.7.1 *WFP #1: Sequence*

Enabling a task after the completion of a preceding task; Figure 6-8.

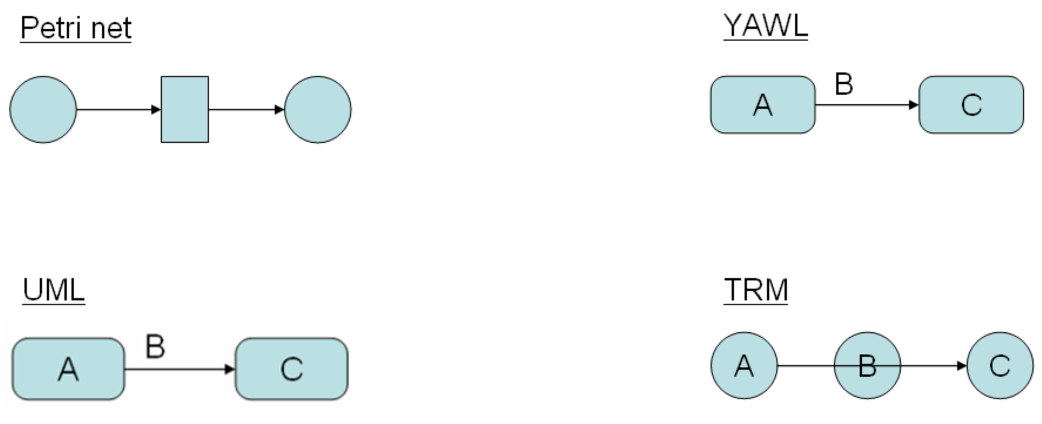


Figure 6-8: WFP #1

6.4.7.2 WFP #2: Parallel Split (AND-split)

The divergence of a branch into two or more parallel branches; Figure 6-9.

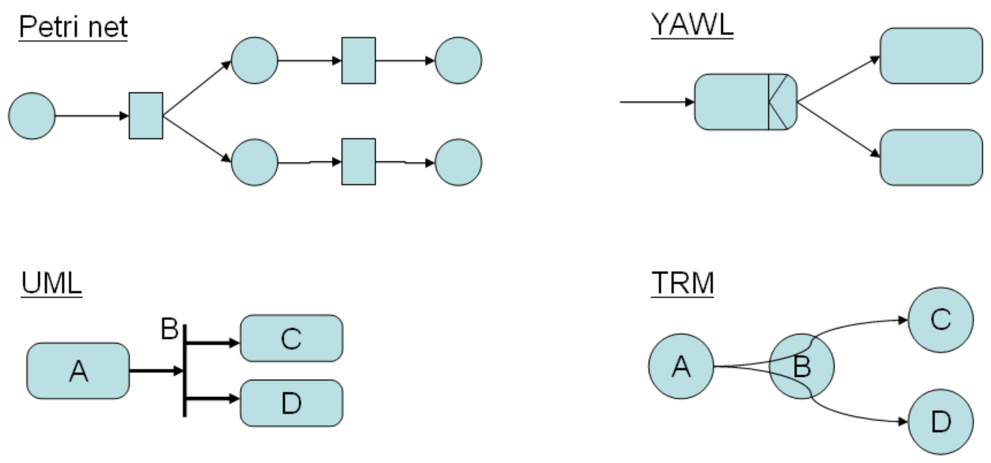


Figure 6-9: WFP #2

6.4.7.3 WFP #3: synchronization (AND-join)

The convergence of two or more branches into a single subsequent branch such that the subsequent branch is enabled when all input branches have been enabled; Figure 6-10.

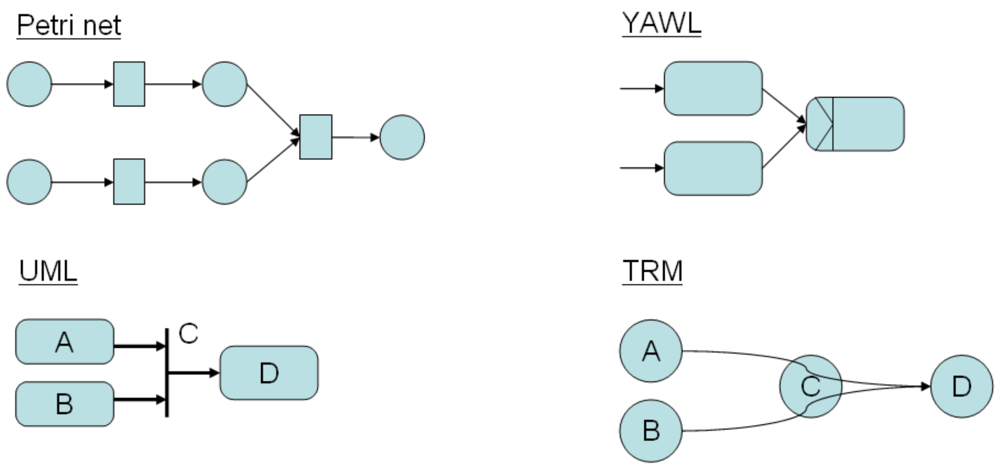


Figure 6-10: WFP #3

6.4.7.4 WFP #4: exclusive choice (XOR-split)

The divergence of a branch into two or more branches such that only one of the outgoing branches can be enabled; Figure 6-11.

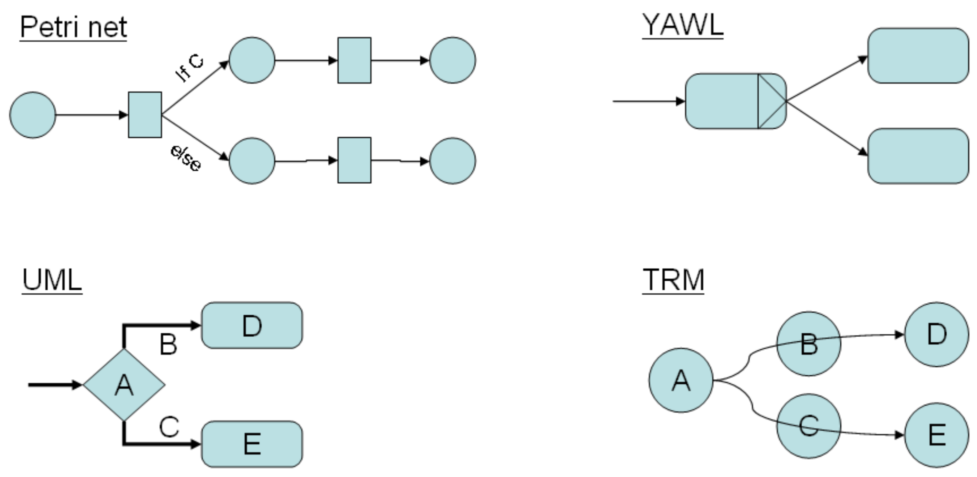


Figure 6-11: WFP #4

6.4.7.5 WFP #5: simple merge (XOR-join)

The convergence of two or more branches into a single branch such that an enablement of an incoming branch is enough to enabling the output; Figure 6-12.

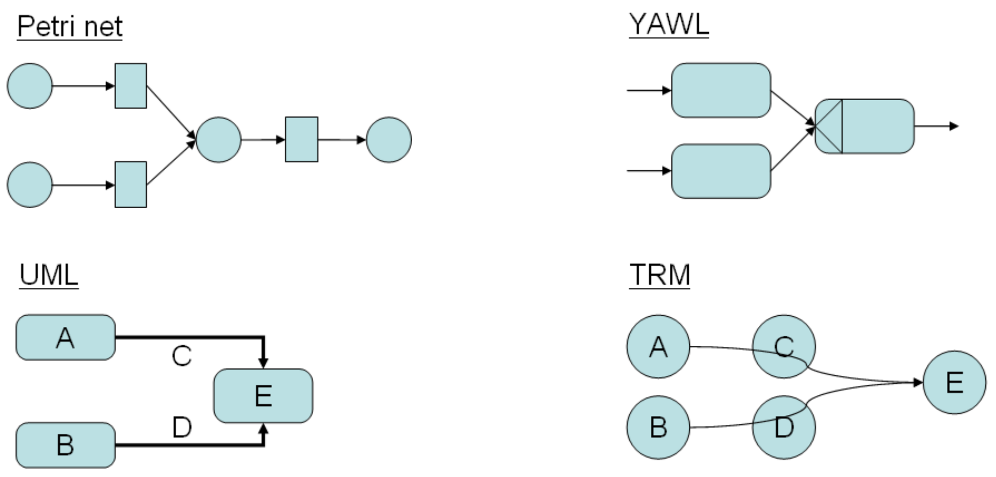


Figure 6-12: WFP #5

6.5 Discussion

Through the comparison done in the previous section, simplicity can be seen as the main advantage of the TRM-WF and the highlighted point is that everything is a "node" in the TRM and complex workflows can be seen as a number of nodes in a TRM space. This is while other notations may use multiple elements in the definition of the same workflow.

The disadvantage can be seen as increased complexity of the TRM-WF graph when the workflow grows. This is because the TRM graph is mostly designed to illustrate the TRM concepts, not to be practically implemented in real problems. A practical TRM-WF data is usually too huge to be always represented as a graph. It is noticeable that the target of the TRM-WF is to simplify the logic of the workflow modelling and not necessarily to simplify its visual illustration.

Another important advantage of the TRM modelling is in re-using of the nodes. As defined before, the TRM allows usage of a single node in several relations as source, destination or association. This is while in other notations, there is no way of re-using an activity in two different locations of the graph.

Finally, the bi-directionality of the TRM is another important distinction over other standards. This advantage is unique to the TRM-WF and cannot be seen in any other studied workflow models. If a relation can be read in two opposite ways, then the question is what is the meaning of a workflow pattern when it is read reversely? The answer comes from workflow management systems, when users need to roll-back an activity. If a user wants to withdraw an already passed work case, the system must provide necessary roll-back mechanisms. This functionality will help to have limited control on sent-items to draw them back into ready-to-study cases. The reverse meaning of a pattern which is provided by the TRM modelling can help the developers of a workflow management system to make it more flexible in such conditions. The use of this feature is demonstrated as two examples in the next section and the next chapter. Another example of the developed system based on this functionality can be found in [151].

6.6 An Example

In this example, Figure 6-7 is recalled and the nodes are replaced by some real tasks: An insurance company has two groups of users: The board and administration. After a claim is received, the board considers it and decides whether to accept or reject the claim. If it is accepted then the board does the necessary arrangement for its payments and the administration will send a letter of acceptance to the claimant. If the claim is rejected, only a rejection letter must be sent to the claimant. The UML Activity Diagram and its equivalent TRM-WF graph are shown in Figure 6-13.

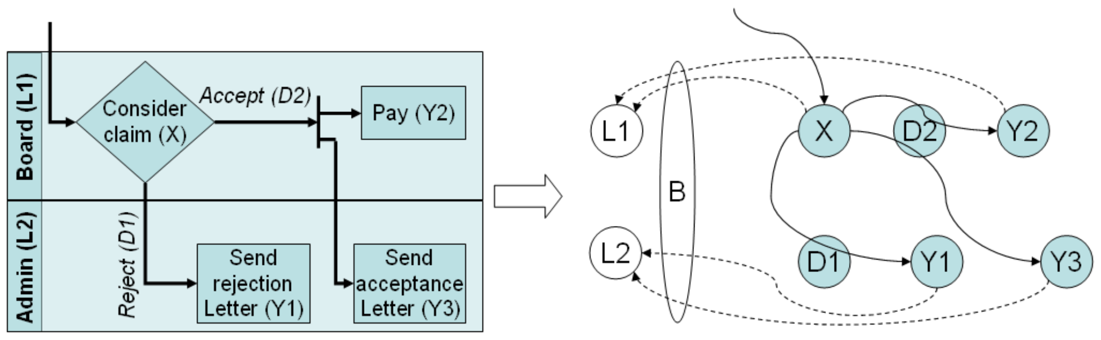


Figure 6-13: UML Activity Diagram and its TRM equivalent for the example of insurance claim

Here the usage of the TRM bi-directionality can be demonstrated. For example, the node D2 has a description like “claim acceptance”. According to the TRM-WF principles, the direct association of D2 is supposed to show “how” a claim is passed from node X to node Y2, something like “the claim is accepted”. From the other direction, the reverse association of D2 must show how a claim moves backward from Y2 to X, i.e. a description of withdrawing a claim acceptance. So the reverse association should have some description like “the claim acceptance is reconsidered”. The same thing can be realized for the node D1. It is also noticeable that the application layer must be equipped to use the bi-directionality features of the underlying workflow model to make this advantage practical.

The drawn TRM-WF equivalent of that sub-workflow can be written in TRM-XML. The following TRM-XML listing is equivalent to the graph of Figure 6-13.

```
<?xml version="1.0" encoding="UTF-8"?>
<TRM xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Research\TRMXML\TRM.xsd">
<node><id>X</id>
        <desc>Considering the claim</desc>
</node>
<node><id>D1</id>
        <desc>Claim rejection</desc>
        <da> the claim is rejected</da>
        <ra> the claim rejection is reconsidered</ra>
</node>
<node ><id>D2</id>
        <desc>Claim acceptance</desc>
        <da> the claim is accepted</da>
        <ra> the claim acceptance is reconsidered</ra>
</node>
<node><id>Y1</id>
        <desc>Sending rejection letter to the claimant</desc>
</node>
<node><id>Y2</id>
        <desc>Payment procedure</desc>
</node>
<node><id>Y3</id>
        <desc>Sending acceptance letter to the claimant</desc>
</node>
<node><id>L1</id><desc>The board</desc></node>
<node><id>L2</id><desc>The administration</desc></node>
<node><id>B</id><desc>Belonging</desc></node>
<node><id>R1</id>
        <src>X</src><asc>D1</asc><dst>Y1</dst>
</node>
<node><id>R2</id>
        <src>X</src><asc>D2</asc><dst>Y2</dst>
</node>
<node><id>R3</id>
        <src>X</src><asc>D2</asc><dst>Y3</dst>
</node>
<node><id>R4</id>
        <src>X</src><asc>B</asc><dst>L1</dst>
</node>
<node><id>R5</id>
        <src>Y1</src><asc>B</asc><dst>L2</dst>
</node>
<node><id>R6</id>
        <src>Y2</src><asc>B</asc><dst>L1</dst>
</node>
<node><id>R7</id>
        <src>Y3</src><asc>B</asc><dst>L2</dst>
</node>
</TRM>
```

6.7 Summary

In this chapter the use of the TRM as an alternative representation of workflows and processes is investigated. The proposed model called "TRM-WF" includes the TRM notation adopted to describe workflows, and the storage layer method (like the TRM-XML) as the language. The Petri net notation, and transitively UML's activity diagrams and the Workflow Patterns have been reduced to the TRM-WF, and it has been concluded that the TRM-WF can represent anything representable in these other models. Moreover, the bi-directionality of the TRM-WF provides more functionality than the other uni-directional models. The simplicity of the TRM-WF in terms of its logic has been compared to the graphical complexity of the TRM-WF in large-sized data. Also the ability of the TRM-WF to describe a modelled set of processes in its own serialised language (TRM-XML) has been shown.

The future works in continuation of this new model and language can be in two different directions: Theory and practice. In terms of theoretical works, the question is about the generality of the TRM model and which other information models can be covered by the TRM, and basically how beneficial is this coverage. In terms of practice, it is necessary to investigate the efficiency of workflow management systems built on this theory. Although the systems described in [151] have been designed based on the TRM theory, the TRM-XML has not been used as the process description language yet.

Chapter 7 -

TWM: A PRACTICAL WORKFLOW DEVELOPMENT

This chapter is dedicated to the development of an online workflow system based on the TRM-WF model explained in the previous chapter. The system is called TWM: "Trm-Wf Management system". In addition to the basic workflow management features, the system is specially designed to demonstrate some of the highlighted features of TRM like the bi-directional and dynamic links, as will be seen later.

The system is designed to be online and web-based in order to avoid any device or platform dependence, in addition to the accessibility reasons. The workflow definition is stored based on the TRM-DB theory in a MySQL database; however, it could be implemented in TRM-XML without any basic change. PHP scripting is used to build HTML pages by accessing the MySQL data and JavaScript is used within the PHP and HTML codes when necessary.

The system has a complete abstraction of workflow definition from execution. This means that any workflow design issue has nothing to do with the coding and all must be applied on the database part of the system. Once the database is fed by the correct workflow definition data, the execution part can operate and provide the appropriate user interface.

TWM's main application is a commercial product that has been installed in several organizations so far. Sections 7.2 describes the practical issues taken in the enterprise-level development of TWM, including user-centred design and an experiment on the users' opinions to support the design. Before that, for demonstration purposes, a simplified version of the system together with a workflow example is described in section 7.1. This demonstration version is available online at http://cs.nott.ac.uk/~axp/workflow.

7.1 TWM Demonstration

Although the demonstration program does not have all of the features of the main version, but it has enough features to support how TRM-WF can be used in designing a workflow management system. The example business process is a customer technical support workflow, namely in a computer service team. Although the example is a relatively simple one, this simplicity is on the workflow definition level and there is nothing in the execution level to stop dealing with the longer workflow definitions.

The involved users in this example are the reception, the helpdesk assistant and the engineer. These three users deal with the technical fault reports from the customers. Briefly, after a customer call, the call is registered by the reception and referred to the helpdesk. The level of expertise of the helpdesk is enough to cover some elementary problems or straight-forward calls, to prioritize the call as regular/urgent or to refer the call to the engineer if more technical help is necessary. The helpdesk is available 24-hour but the engineer is not, so prioritizing may help to refer the calls to the standby engineer in off-hours only if the call is really urgent. The engineer firstly tries to fix the system by remote means, and if the engineer attempt is not successful then a visiting appointment will be set by the reception. In all of the above cases, the call must be referred to and closed by the reception. The above process is illustrated in Figure 7-1.

The dotted line surrounding the “Time?” decision box is not a standard UML notation, but it has been used in the domain of TRM-WF to represent dynamic or functional links. Unlike the other man-made decisions, here the decision between being “business time” or “off-time” is done by an internal system function, as simple as a program that checks the current time. In a more complex example, this program can be a function of workflow nodes or other environmental parameters, as explained in section 3.1.2.

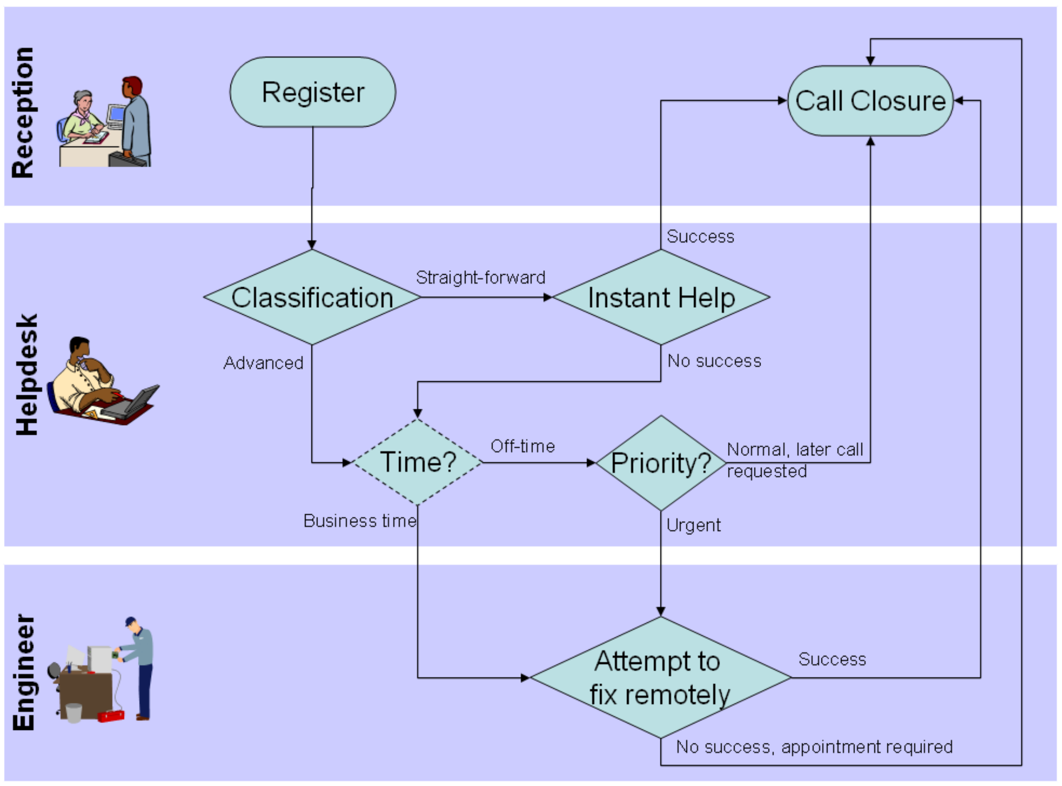


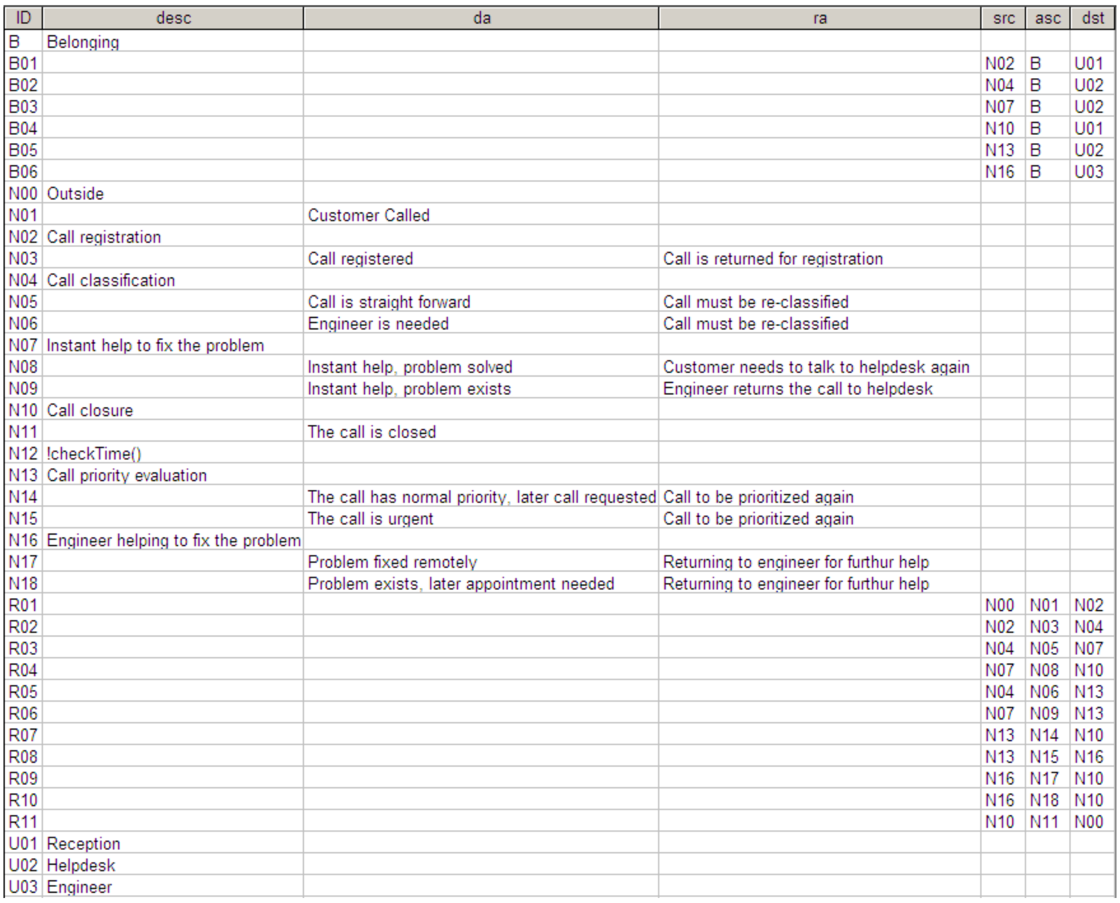
Figure 7-1: The activity diagram of the TWM example

In order to demonstrate the bi-directionality properties of TRM-WF, each relation defined between two workflow boxes includes all the necessary information for the meaning of forward or backward link. For example, not only the reception can refer a call to the helpdesk with the result of "call registered", but also the helpdesk can refer the same case back to the reception notifying that "the call is not registered properly" or "call is returned to re-register", etc.

There are also some other TRM unique properties that are not applicable in TWM. For instance, it is possible for a TRM node to be source of a relation and association of another relation in the same time, but this concept cannot be mapped into workflow, at least at the level of this example.

7.1.1 *The Information Structure*

According to the TRM-Table theory (section 4.2), a single global-schema table must handle all of the required information of workflow definition. Figure 7-2 shows the table which is designed to represent the mentioned PC Service workflow and is stored in MySQL database.

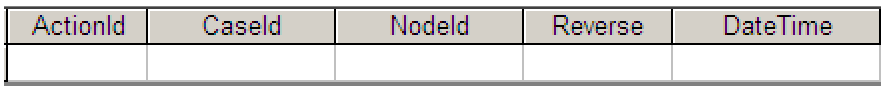


ID	desc	da	ra	src	asc	dst
B	Belonging					
B01				N02	B	U01
B02				N04	B	U02
B03				N07	B	U02
B04				N10	B	U01
B05				N13	B	U02
B06				N16	B	U03
N00	Outside					
N01		Customer Called				
N02	Call registration					
N03		Call registered	Call is returned for registration			
N04	Call classification					
N05		Call is straight forward	Call must be re-classified			
N06		Engineer is needed	Call must be re-classified			
N07	Instant help to fix the problem					
N08		Instant help, problem solved	Customer needs to talk to helpdesk again			
N09		Instant help, problem exists	Engineer returns the call to helpdesk			
N10	Call closure					
N11		The call is closed				
N12	!checkTime()					
N13	Call priority evaluation					
N14		The call has normal priority, later call requested	Call to be prioritized again			
N15		The call is urgent	Call to be prioritized again			
N16	Engineer helping to fix the problem					
N17		Problem fixed remotely	Returning to engineer for furthur help			
N18		Problem exists, later appointment needed	Returning to engineer for furthur help			
R01				N00	N01	N02
R02				N02	N03	N04
R03				N04	N05	N07
R04				N07	N08	N10
R05				N04	N06	N13
R06				N07	N09	N13
R07				N13	N14	N10
R08				N13	N15	N16
R09				N16	N17	N10
R10				N16	N18	N10
R11				N10	N11	N00
U01	Reception					
U02	Helpdesk					
U03	Engineer					

Figure 7-2: Using TRM-DB to express the workflow definition of the PC Servive example

In this table, "ID" specifies the unique node identifier, "desc" shows the node's description, "ra" and "da" define the direct and reverse association of the node, and three fields of "src", "asc" and "dst" are about expressing three element of a relation. According to TWM, "U**" nodes represent the users, "N**" nodes describe workflow tasks (also N00 for the virtual "outside" node), "R**" nodes define the relations between N nodes (equivalent to the arrows in the diagram), and "B**" nodes are about the "belonging" relation between N nodes to U nodes (or simply which task is done by which user). The node "B" itself is the special relation of "belonging" which must be used as the association in “B**” nodes. Finally to represent the functional links, an exclamation mark followed by the name of the function is written in the “desc” field. (like N12).

TRM-DB is used only for defining the workflow. TWM uses another table for the execution of the workflow. The structure of this table is application-dependant and according to the minimum requirements of TWM demo, the design of the execution table is show in Figure 7-3.



ActionId	CaseId	NodeId	Reverse	DateTime

Figure 7-3:TWM execution table

In this table, each action (like passing a case between two nodes) is represented as a row with a unique ActionId. CaseId specifies the passed case, NodeId determines which node in the definition table (TRM-DB) is equivalent to this passage (normally R** nodes) and “Reverse” is a yes/no field that specifies whether this passage is forward or backward through NodeId. Finally the date and time of the passage is stored in the appropriate field.

7.1.2 The User Interface

The general look of the online user-interface of TMW is shown in Figure 7-4.

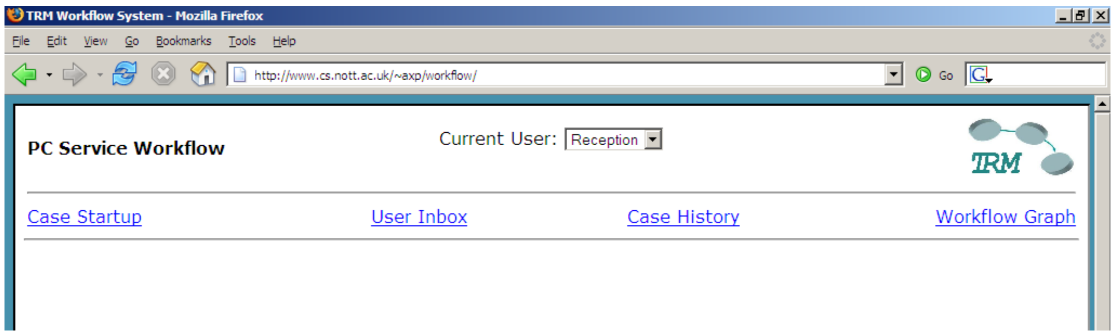
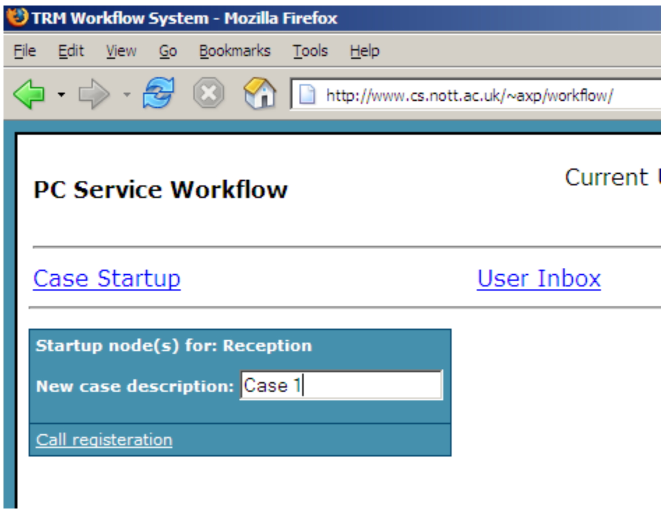

TRM Workflow System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.cs.nott.ac.uk/~axp/workflow/
Go
PC Service Workflow
Current User: Reception
TRM
Case Startup
User Inbox
Case History
Workflow Graph

Figure 7-4: General look of the TMW user interface

In this interface, a drop-down list identifies the user working with the system. Here for simplicity, the authentication processes has been omitted. A user, like reception, can do three main tasks: Starting up a case, controlling his/her inbox, or checking the history of a case. The final option (workflow graph) shows the activity diagram of the workflow to the user, a picture like Figure 7-1.

By selecting "Case Start-up", the system will look at the workflow definition (table of Figure 7-2) to identify the nodes that the current user is able to start a case. Here for the reception user, the only start-up node is N02 (Registration). This can be done by executing a SQL query that looks for the destinations of the relations starting from N00. Starting-up may also mean creating a new workflow case, so the description (or identifier) of the starting case must be entered.

A typical reception starting-up screen is like Figure 7-5. In this figure, the user by typing "Case 1" and clicking on the "Call Registration" will start the workflow of that case.


TRM Workflow System - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.cs.nott.ac.uk/~axp/workflow/

PC Service Workflow

Current

Case Startup

User Inbox

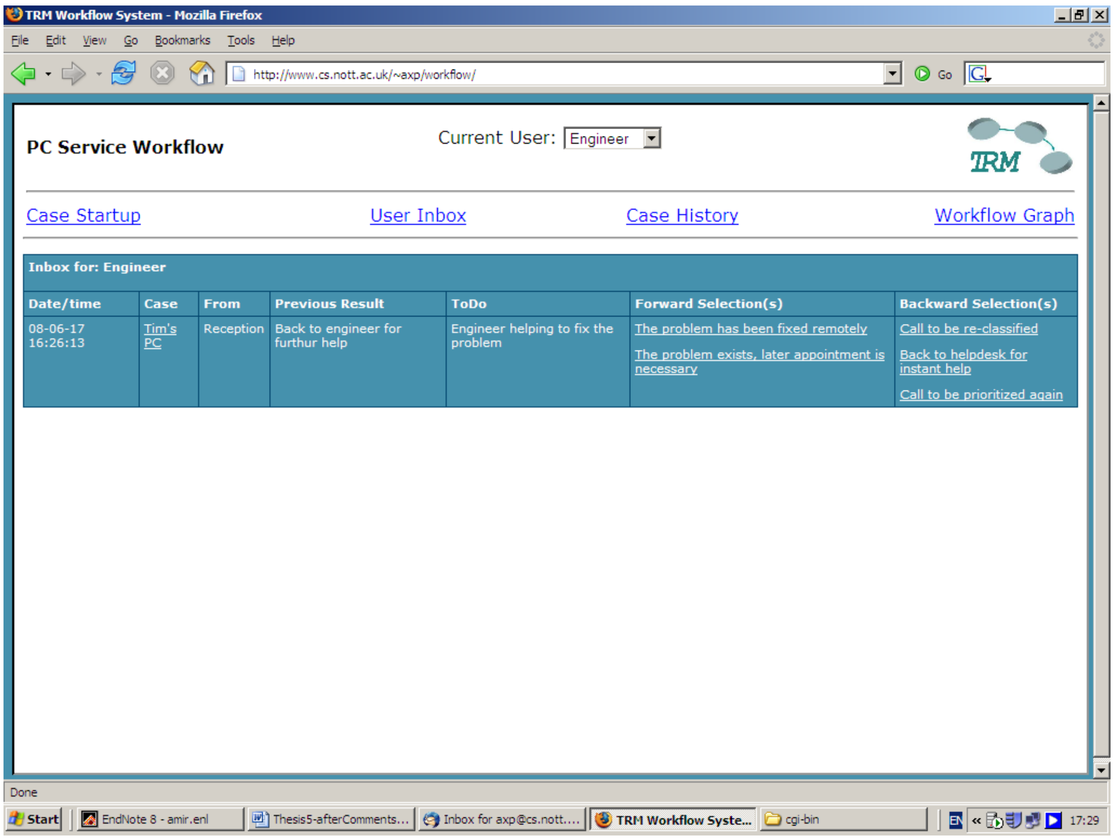
Startup node(s) for: Reception

New case description: Case 1

Call registeration

Figure 7-5: A sample of case start-up screen

By selecting "User Inbox" a table shows the list of tasks that are waiting to be processed by the current user. For example, the Engineer may see Figure 7-6 as her/his inbox.


PC Service Workflow

Current User: Engineer

Case Startup | User Inbox | Case History | Workflow Graph

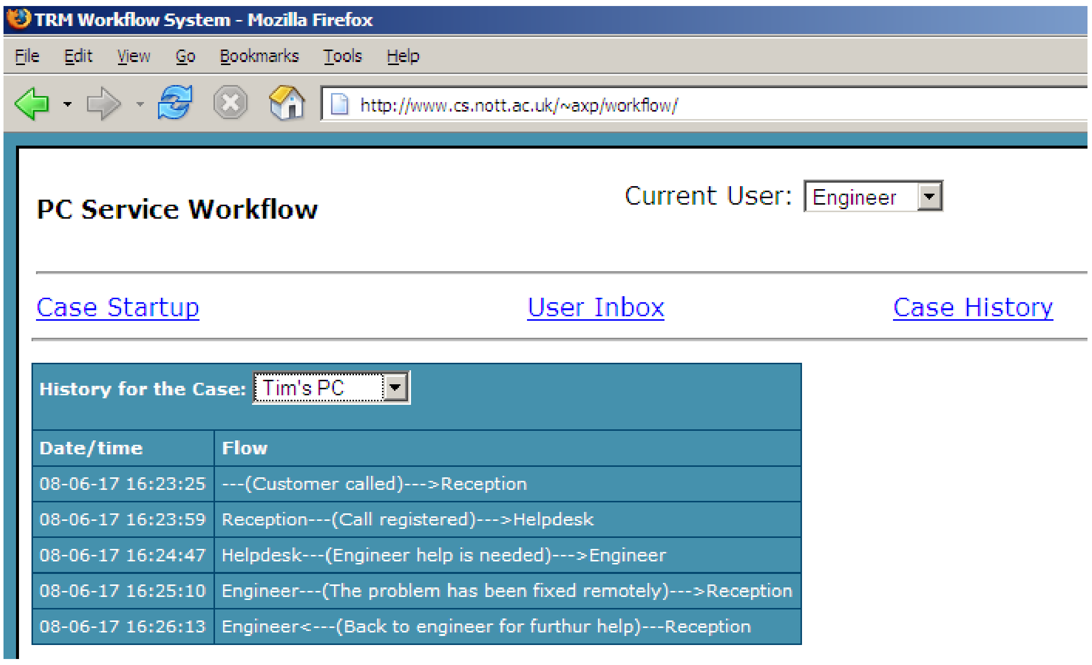
Inbox for: Engineer

Date/time	Case	From	Previous Result	ToDo	Forward Selection(s)	Backward Selection(s)
08-06-17 16:26:13	Tim's PC	Reception	Back to engineer for furthur help	Engineer helping to fix the problem	The problem has been fixed remotely The problem exists, later appointment is necessary	Call to be re-classified Back to helpdesk for instant help Call to be prioritized again

Figure 7-6: A sample TWM Inbox for user "Engineer"

Hyperlinks in the last two columns are doable tasks, including tasks to pass forward or backward. The user may have zero to many rows in his/her inbox, for any row he/she may also have some decisions to pass the case forward, and/or some decisions to pass the case backward. The look of the link specifies the result of the action when it is clicked. So for example if "Back to helpdesk for instant help" is selected by the Engineer, then the Helpdesk will see that text in his "Previous Result" column. This screen has been specially designed to show the practicality of the TRM theory by feeding this table from TRM-DB, which includes all of the required information for the user's decision making.

By selecting "Case History" the user can see a list of all the actions done for a specified case. For example, a sample case history may be like Figure 7-7.


TRM Workflow System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.cs.nott.ac.uk/~axp/workflow/
PC Service Workflow
Current User: Engineer
Case Startup
User Inbox
Case History
History for the Case: Tim's PC

Date/time	Flow
08-06-17 16:23:25	---(Customer called)--->Reception
08-06-17 16:23:59	Reception---(Call registered)--->Helpdesk
08-06-17 16:24:47	Helpdesk---(Engineer help is needed)--->Engineer
08-06-17 16:25:10	Engineer---(The problem has been fixed remotely)--->Reception
08-06-17 16:26:13	Engineer<---(Back to engineer for furthur help)---Reception

Figure 7-7: A sample Case History screen in TWM

In this screen, left-to-right arrows mean passing forward and right-to-left arrows mean passing backward. Also the look of each passage represents the TRM notation, i.e. a relation starting from a source node, passing through an associative node, and terminating in the destination node. Moreover, the text showed for the association node (surrounded by brackets) is either the content of field "da" (for the forwarding passages) or the content of field "ra" (for the backward ones). This screen has been also designed to demonstrate the practicality of the TRM theory and notation.

7.2 The Enterprise-scale Development Experience

This section contains some of the major design considerations which have been experienced through development of TWM for organizations and offices. These considerations, which are raised from both theoretical and practical sides, include the gathered requirements in a user-oriented iterative design, the implied changes in software architecture to satisfy these requirements and the developers' experiences on how such workflow systems can be easily adopted in typical office environments.

The enterprise edition of the TWM can support defining and executing multiple workflow systems in an organization having various groups of users. The system has been tested to support 100 active users on a single server; however the number of users is by no means limited to that unless the server or the database has such a limitation. As will be studied later, having a user-centred design and managing to satisfy the theoretical and practical requirements for developing a workflow system are the main highlights in developing the TWM. Gathering the user's requirements before and in parallel with the development as well as recollecting the users' feedbacks and iterating this cycle has been done through verbal and written communications with the users in different stages of the development.

More specifically, feedback from 40 users about the desired functionalities of a workflow system after using it has been compared with the initial requirements gathered before the development. This comparison shows not only the essential users' requirements from a workflow system, but also how some of the requirements can be changed through a cyclic design.

7.2.1 The Usability Issues in WFMSs

Workflow management systems are one of the important enterprise applications. The design of sustainable enterprise applications requires much focus on the usability issues. Iterative and user-centred development methods are known approaches to make such systems more user-friendly and sustainable. In these methods, users of the systems are not only those who have ordered a system (like in the Waterfall Model of software engineering [158]), but those who are highly involved in the designing processes. Iteration here means that feedbacks from users about the designing software, as a whole or as a part, are used to re-design the system. [23].

While there are hundreds of pre-designed workflow management systems being used, still many organizations need customized workflow applications to be specially designed for them concerning their special needs [194]. Here it is tried to show the implications of user's requirements on the development of a workflow system through an iterative design and to show how such a design may address these requirements. Also it will be shown that experiences of developers may be used to predict functionalities that the invoice user may not consider in the first stages but may wish for later.

It is observable in the workflow literature that researchers put mostly emphasis on the theory of workflow modelling and most of the efforts are devoted to technical issues or abstracted workflow modelling [52]. This is while users may have a different class of concerns that may be missed in an isolated software design. Through development and implementation of several real-world workflow systems, it has been concluded that a main factor in making sustainable workflow systems is an optimal balance between technical and practical sides of development [52]. This research tries to share the lesson learned when such a balance is targeted.

Another point that has been observed in organizations during this research is the existence of changes in users' expectations after deployment of workflow management systems. This can be considered as a part of the organizational changes enabled by workflow systems [163].

7.2.2 The Related Works

Beside the related theoretical works mentioned in sections 2.7, there are some other related works that are focused on combining the theory with the practical side of the workflow technology.

ADEPT [65] is a complementary framework that tries to cover more theoretical and practical sides of workflow modelling by providing more possible concepts and actions in such systems, like systematic addressing the pre-planned exceptions in order to adequately capture real-world processes (e.g. forward and backward jumps), ad-hoc derivations from the pre-modelled workflows, covering inter-workflow dependencies, advanced user interface, and some trends to optimize enterprise-wide communications. Rollback is another systematic concept that has been added to the workflow model, which has not been completely covered by the classic models. Also ADEPT has been used as a basis in other development research projects like AristaFlow [18].

Management of workflow systems while spreading them to enterprise-wide applications can raise some other concerns in software architecture that again may not be fully addressed in the classical models or in many commercial products [13, 42]. End-user access tools, workflow modelling tools, workflow instant management and project planning tools are different fields in the optimization of a workflow management system in order to make the product more sustainable in such scales.

Different stakeholders in workflow community, i.e. academics, vendors, organizations and users, can raise different expectation from a workflow management system. Through synchronization of these stakeholders' expectations, some researches verify that the general results meet data from the theoretical side [109], while interestingly some others mention that the theoretical side of current workflow products are quite unprepared to meet the practical users' demands [13].

This balance must be reassessed independently for each certain workflow application by putting users at the centre of the design. This is why the development process of a workflow management system (which is itself a workflow) is another subject of research. Although a few works have been devoted on this area, a reference workflow application development model introduced in [194] is built on real-world experiences. In this model, the involvement of the empirical studies, gathering users' requirements, business process modelling and workflow modelling into the design processes have been studied.

Although it is observed that the details of the user-desired functionalities or features are not a matter of interest for the theoretical researchers, authors of [9] have shared some detailed experiences in implementing a workflow management system. The studied features in that work, which may consequently affect the design principles, include explicit process definition tools, process enactment facilities, tracing tools, monitoring and reporting tools. The lessons they learned in their experiences are close to the lessons learnt in this section. The user's feedback on using the system was not uniformly enthusiastic in their research and they experienced negative feelings when the users were presented with some electronic versions of their previous paper forms. Another negative users' feedback mentioned in their work is about the workflow definition tools when users need to redefine or modify the workflow. This problem comes back to this fact that many workflow definition tools or definition languages (like WPDL [195] or XRL [8]) may need a certain level of computer knowledge, which normal users may not have. The later point can raise many usability issues that may be addressed by introducing graphical or textual workflow definition tools. In the field of textual tools, easy process description languages which are close to natural languages (like in [149, 152]) can help users in these issues.

The encountered problems that developers of workflow systems have experienced are almost same in nature. A comprehensive set of those problems has been reviewed in [194], like isolation of technical from organizational aspects, development without prototyping, unsuitable transfer of paper works to automatic processes and server performance issues in the enterprise-wide applications.

7.2.3 Methodology

The method used in this research includes gathering initial users' requirements, using these requirements in the software architecture design, and gathering users' feedbacks after short-term and long-term operational phase. The development method follows the iterative design, in which the users are highly and actively involved in a cyclic process to test the system and share their views with the developers. The developers are also asked to validate the implemented system from the users' perspective.

This method has been used in four offices of different types with different businesses: 1)TV production process in a TV program production organization. 2)Office works of a government-affiliated charity to help homeless people 3)Business processes of a multimedia advertisement company; and 4) Workflow of an international conference management company.

A total number of 40 active users in those four businesses have been selected for answering two similar questionnaires before and after using the system. The first set of questionnaires has been answered by them before starting development and the second set has been answered after having long-term (2 years) experience in using the developed system. It is also noticeable that none of these users have any previous experience with a real computerized workflow system.

The result of the experiment provides general guidelines and advices for future workflow system developments. The answers to each question are focused almost separately and the changes in number of different answers to each question can roughly lead to certain conclusions. Thus the no statistical analysis is necessary.

7.2.4 Gathering Initial Requirements

In the initial step, general requirements of top users (or managers) are very important to be gathered. Managers in this step are usually interested to replace their manual system with a computerized workflow system, while having some true or false image about the functionalities and benefits of such systems. These main requirements include:

1- They usually have set of graphical flowcharts that need to be fed into the new system as the raw material of workflow definition. However, these flowcharts must be re-engineered in many cases.

2- The system must be able to direct users to do what they are supposed to do, regarding the workflow definition.

3- From a managerial point of view, the system must be able to show and trace history of processes for each case, and to show the details of each user's actions to certain classes of users.

The above general requirements have been studied in depth by the developers and system designers to reach the detailed specification. After more discussions and brain-storming sessions, some more specifications of the system have been shared between the developers and the users, like:

1- The system must have enough flexibility to accept some frequent changes on the workflow.

2- There may be several workflows in a single organization, with or without gateways between them, while a single system is supposed to manage them together.

3- There may be several differences between the workflow designed for manual system, and those who must be used in the computerized one.

4- There may be some data forms that the users are supposed to fill before passing the work case to the next node.

5- The existence of a messaging system between users while passing the work case seems necessary. This can also be classified as public or private messages.

6- If a user wants to withdraw an already passed work case, the system must provide necessary mechanisms.

7- The workflow is defined to apply to roles (or jobs) of the users, not to users themselves. Each user may have a different role in respect to each workflow case.

7.2.5 Functionalities and Implications

Based on the requirements described above, the main functionalities that need early design concerns have been extracted as different 'forms' in the system's user interface, as follows:

1- *Ready-to-Study Cases Form:* A form is needed to be designed that contains all of the cases that the current user is supposed to do, i.e. those which are waiting to be studied by the current user and passed to the next one. This will look like the "inbox" folder in email clients. It also must contain the detailed information about the previous study which has been done on the work case by another (or same) user. For studying a case, the user may or may not fill a data form or message to the next user. Also a confirmation about the next destination of the work case will be shown to the user before passing the work case.

2- *Sent-items Form:* A form is needed to be designed that shows a part of sent-items which are not passed to a third party. These items are exactly those which are able to comeback to the ready-to-study cases form, if the user wishes to do so. There may be repeated items with different destination, if the current user has passed a work case through a distribution node. In these cases, drawing one of them back means withdrawal of all of them, and having them in the ready-to-study case as a single item.

3- *History and Current-status Forms:* These two forms must be designed, preferably within a single user interface, to show where were and where are the moving work cases in the defined workflow. This has more importance from a managerial point of view to trace and investigate the stops and movements of the work cases. In some cases, these two forms may show the move of work cases from a workflow to another, if the system provides such inter-workflow jumps.

4- *Workflow Definition/Change Interface:* This form must provide the functionalities to design, review, change and update the workflow definition by certain classes of users. There are two possible methods about how to manipulate the workflow definition: textual and graphical. In the textual mode, a workflow definition language has been used. This has been called PDL (Process Definition Language) and it is a very simple language, similar to the structured English and can be read and understood by normal users. A parser converts the lines of PDL to a set of SQL statements that can be used to define or change the data in the definition layer of the database. More details on PDL and its implementation can be found in [149].

5- *Workflow-bypassing Form:* For escaping from happening deadlocks, or for addressing many practical issues that may happen in offices, some certain classes of users must have access to this form, which is designed to bypass the defined workflow. It provides the facility that the user can pass a work case from the current node to some other node that the defined workflow doesn't allow directly. This may practically include jumping over nodes, cancelling or taking a work case away from a certain node or acting on behalf of another user.

6- *Withdraw Form:* As a part of exception handling, withdraw forms are necessary to be designed. These forms will help to have limited control on sent-items to draw them back into ready-to-study forms.

7.2.6 The System Architecture

Concerning the above features, the main items about the system architecture have been concluded as:

1- The system is based on client-server architecture with a central database. The centralized database has been selected considering the size and scope of the workflows and organizations.

2- The information stored in database includes two abstract layers, named "definition" and "execution" layers. This abstraction also complies with the Workflow Management Coalition reference model [195] when two different gateways are considered for definition and execution. "Definition" is the lower layer which contains all information about the definition of a workflow, and "execution" is the upper layer which contains all information about workflow cases and all processes which have been done on each task by users through the defined workflow in the lower layer. Although these two layers are dependant, this abstraction gives the system more flexibility in terms of accepting the workflow changes.

3- The required flexibility of the system in terms of accepting the frequent workflow definition changes must consider keeping the execution layer information (which may be based on old definition data) always safe, integrated, valid and usable. This must be done by predicting database support to these changes.

4- The system may ask users to fill a data form associated to each node, when they want to pass a work case from that node. This implies having sub-databases for manipulating data in each data form. This also implies conjunction of the workflow database with a document management system.

5- The method of converting drawn graphical flowcharts to the information stored in the definition layer is an important stage. Some kinds of process engineering expertise is needed in this conversion, since a complete understanding of the organization and its process is necessary, as well as understanding the future plan for computerizing the system. This implies graphical flow-charting tools and/or special language parsers to joint to the whole system.

7.2.7 A Research on User's Requirements

A number of 40 users from different organizations have been selected for this research and they have been initially asked to answer a questionnaire. It is noticeable that none of these users have a real experience with any computerized workflows before this research. This has been done just before designing the general specifications. The research has been repeated with the same questionnaire for 40 active users (including 3 replaced persons) after 2 years from the first user trials. During these two years the system was operational and had been used actively by these users.

The result has been summarized in Table 7-7-1 and Table 7-7-2.

Table 7-7-1: Results of a comparative research among 40 users (part 1).

Issue	Options	Stage 1 (before)	Stage 2 (after)
1. How do you like the workflow system to limit the users in their office works	a. No restriction: Such systems are to answer the informational requirements of users, not to restrict them	14	5
	b. Passive: Such systems must show the users what to do, but not restrictive	15	12
	c. Active: Such systems must limit the users to do their office works in the right direction	11	23
2. The desired method for withdrawal of a work case after being passed	a. Should be impossible	9	2
	b. Users can always withdraw unwanted passing	21	12
	c. Users can withdraw unwanted passing only if the next user hasn't passed it.	10	26
3. The interface for the studied cases (sent items) must:	a. Show all the sent items	15	13
	b. Show those who are ready to study by the next user	25	27
	c. Highly restricted to special users	0	0

Table 7-7-2: Results of a comparative research among 40 users (part 2).

Issue	Options	Stage 1 (before)	Stage 2 (after)
4. Access to the history of work cases' passing	a. Show all the history	15	21
	b. Show what the current user has done	22	14
	c. Highly restricted to special users	3	5
5. Access to the current status of work cases	a. Show all the current stop points for a work case	16	28
	b. Show those who are for this user	19	4
	c. Highly restricted to special users	5	8
6. Tools for workflow definition and changes	a. Essential	5	26
	b. Good	31	10
	c. Redundant	4	4
7. Graphical tools for workflow manipulation	a. Essential	12	8
	b. Good	25	24
	c. Redundant	3	8
8. Possibility of workflow bypassing	c. Must be impossible	18	7
	a. Highly restricted to special users	12	19
	b. Must be available in some extent to all users	10	14

7.2.8 *Discussion and the Lessons learnt*

1- Answers to question 1 about the desired level of general restrictive behaviour of workflow systems shows how restrictions can be accepted and even increase user satisfaction, when it is used in the right way. The point is that the general understandings of the restrictive behaviours of the computer systems to the users are shaped when the system limits them to do what they are supposed to, but in a well-designed workflow system, this can be converted to satisfaction if they find the system allowing them what they are supposed to do and denying them otherwise.

2- The mechanism for withdrawal of the passed work case (question 2) are mostly desired to be applicable in all situations, whether the next user has assed the work case to a third party or not. This look has been moderated in the second stage. People now mostly like to have withdrawal capabilities if the next user has not passed the work case anywhere. This is partly because of practical problems that might be caused by the free withdrawal method.

3- Answers to question 3 about "sent-items" folder are almost the same in both stages. Users like to have a folder called "sent items" but they mostly like it to contain the ready-to-study items, not all items. This is because they prefer to have the choice of seeing the history of each work case using the history screen, but in the sent-items screen they prefer to see the items which they can withdraw.

4- Answers to question 4 about the history page have changed over time. This shows that users imagined that it is enough if they know what they themselves have done in the past about a certain work case, but after experiencing the system, they feel more interested to know all the history about it.

5- Similarly, the above conclusion can be said about answers to question 5 about the current status of work cases.

6- Answers to question 6 about workflow definition tools show how important the existence of these tools is. People in stage 1 had no clear idea about how frequent the changes on the workflow definitions are, or may have thought that such changes are easy to apply without specific tools. This view has been corrected in the second stage.

7- Graphical tools for workflow manipulation (question 7) were an attractive idea for users in stage 1, but not so much in stage 2. This shows that in a busy office, users may have not enough time to use a graphical tool, or the textual information had enough functionality for them.

8- The answers to question 8 about the possibility of bypassing work cases show more restrictive views of users in stage 1 than in stage 2. This also can show that the practical situation that they may have encountered in the past has guided them to consider more flexibility of the system in terms of workflow bypassing, at least for some certain classes of the users.

7.2.9 What does the experiment mean to the TRM

As introduced earlier, the experiment has been done to make a user's centred design, so not all of the covered items are necessarily dealing with the technical issues of making the system. Also the initial ideas of making the system had come from a TRM centric design. As concluded in the previous chapter, the TRM provides the theoretical platform to add two main features of "functional links" and "bi-directionality" to the classical workflow systems.

Although user's cannot explicitly observe any TRM-aware issues neither in the questionnaire nor in working with the system, but the results of the experiment has interestingly support the TRM added values: Questions 1 and 8 require having the functional links and question 2 requires having the bi-directionality features in the data model layer of the developed system. These features are not necessarily impossible without having an explicit TRM approach in design, but such an explicit approach can make the development more robust and consistent.

7.2.10 Conclusion

Observing a gap between theoretical modelling and real-world practical problems, it was tried in this section to make a balance between these two sides by more focus on the details of the generally-desired workflow features. These experiences also shows how the users' requirements before and after using a system can be re-used in the iterative designing stages and how the architecture of a system must obey them. The role of the developers in predicting the future users' requirements has also been focused, so the users' wishes can approach to the developers' ideas. Finally, this approach can give more operational sustainability to a working management system.

7.3 Summary

This chapter studied the possibilities of practically applying TRM-WF model in developing a workflow management system, TWM. For this purpose, two approaches have been taken. First, through demonstrating a simple developed TWM, it has been shown how TRM-WF can be used to design the structural information model, how the system can use the information stored in the TRM-DB to manage the system, and build different user interfaces. In the second approach the lessons learnt during the extension of the TWM in an enterprise level has been shared, concluding that a balance between the theoretical and user-oriented design must be present to make the system more sustainable.

Chapter 8-

DISCUSSION

In this chapter, the TRM theory and practice developed in the previous chapters will be revisited briefly to integrate all of this research. The TRM will be evaluated against the stated objectives and aims, other work areas will be discussed, and the abstracted fundamental underpinning the ternary nature of the TRM will be considered.

8.1 Revisiting the Objectives

According to the objectives of this research (section 1.3), the study method includes top-down, formation and bottom-up steps. The top-down method was designed to collect all the evidence and requirements for forming the targeted model, and the bottom-up used the TRM to construct new information top models.

It is useful to revisit the progress of achieving the objectives as a whole. This progress has been shown in Figure 8-1.

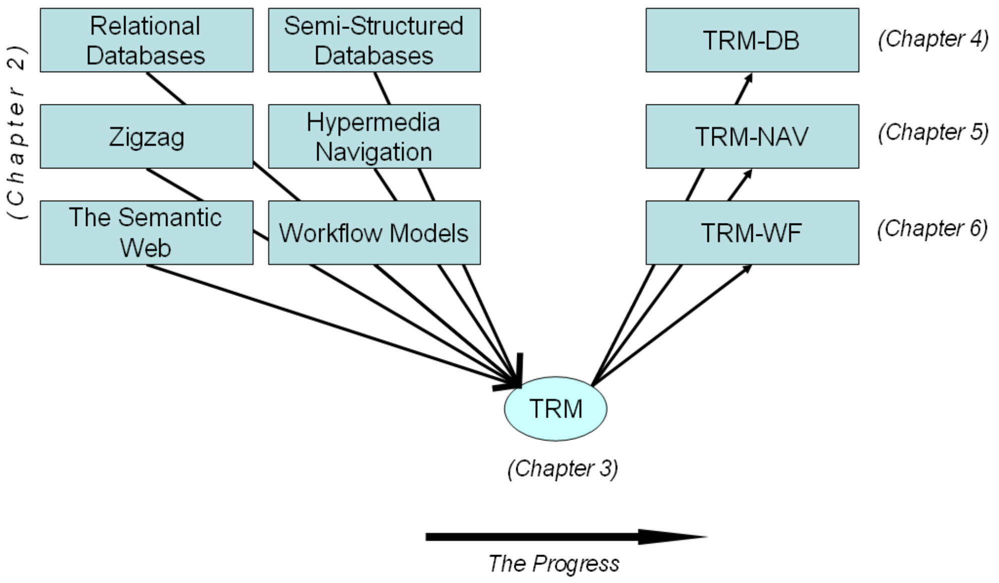


Figure 8-1: The progress of the development of the TRM

8.2 The Forming Ideas

The above top-down method has led to forming the TRM mainly in chapter 3, but the ideas behind this formation have not been explicitly focused, because it was not possible to do so until the end of the bottom-up method. Let us revisit the formation of the TRM by another basic approach, which may help reaching some new conclusions.

What inspired the development of the TRM, as with most abstract models, was a combination of *evidence* and *requirements*. The evidence was taken from the commonalities between related works (chapter 2). This has made it apparent that there is usually an extractable relativity between three nodes of data. This concept of "triples" is important because these relationships require grouping into threes – neither more nor less. The requirements originated from studying the differences between the related works and from observing some real-life needs in the studied models. Studying the requirements was important because it could make it clear that the TRM should have properties such as simplicity, dynamicity, bi-directionality, flexibility, etc.

The optimality and the effectiveness of the TRM have been shown to be dependant on both evidence and requirements, as illustrated in Figure 8-2.

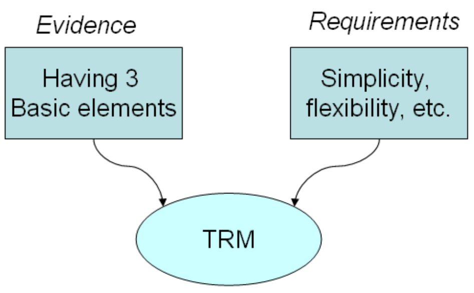


Figure 8-2: The role of evidence and requirements in forming the TRM

This fact has been supported by the reviewed related works. For example, RDF [184] is an absolute ternary-based model, but this is not enough according to the above approach. As shown in chapter 5, RDF does not support functional links and is not as simple as a single class hierarchy. Thus RDF was put on top of the TRM layer to be shown that it uses a subset of the TRM as its fundamental information layer (the layered approach of section 3.3). Similarly, the BRM [19] is an absolute dynamic model that supports the functional link, but it lacks the first issue, which is being ternary. The same analogy may be used for ZigZag [129], which is ternary, but restrictive.

The TRM was deliberately formulated in a context-free environment to satisfy both the evidence and the requirements (chapter 3). This means that the TRM was developed by looking at what other models may have or may lack, not by looking at the requirements of an individual application.

The union of the requirements has led the TRM to include the following properties:

1. Simplicity, particularly by the rule of "nothing but node", to satisfy the analysis requirements (section 3.1.1).
2. Dynamic links, bi-directionality and avoiding rigidity, to satisfy the practical requirements (section 3.1.2).

8.3 Revisiting the Unification Aim

The first aim of this research mentioned in chapter 1 was to have a unification approach to different related systems. As a conclusion, the TRM has been shown to be able to make a general ternary node-link structure. More specifically the following unification facts were justified in chapters 2 to 6:

1- Relational Databases [58] are shown to be simplified as linking data in some row-data-column triples, thus it is a special case of the Static-TRM (section 4.1.1).

2- Different hyperlink models and methods (like Dexter Model [92], BRM [19], HTML links [183], Spatial Hypertext [167], Process-oriented Model [35], WebML [46], Structural Computing [135, 136], MMVP [17], Metalevel links [172], Trellis Model [169], FOHM [64, 119] and XLink [185]) were shown to be reducible to node-context-node, node-type-node, node-semantic-node etc. (section 2.4.6), all are reducible into special cases of the Dynamic-TRM.

3- XML [185] is shown to be convertible to directed labelled graphs [175] (section 2.2), and consequently to some node-edge-node triples, having no dynamic structure and bi-directionality, thus it again falls into a special case of the Static-TRM (section 0).

4- RDF [184] is based on some object-predicate-subject triples, having a different class hierarchy for each element and no dynamic structure, thus it falls into another special case of the Static-TRM (section 5.7).

5- ZigZag [129] is shown to be reducible to some cell-dimension-cell triples having fundamental ZigZag restriction of linear ranks, no dynamicity and bi-directional, thus it is a special case of the Static TRM (section 0).

6- Different known workflow definition models, languages and notations (Petri net [145], UML [70], BPMN [138], XPDL [196] and YAWL [4]) are shown to be simplified as a set of task-action-task triples with possible dynamic characteristics, but no task can be action and vice versa, thus it is another special case of the Dynamic-TRM (section 6.4).

An overall summary comparing the TRM with the related work is shown in Table 8-1.

Table 8-1: The summary of comparing TRM with the related works

	Ternary relations	Definition Simplicity	Structure flexibility	Bi-directionality	Dynamic links
RDB	implicit	low	no	yes	No
XML	implicit	medium	yes	no	No
BRM	no	high	yes	yes	Yes
Classical HT links	no to implicit	high	yes	no	No
Adaptive/open HT links	implicit	high	yes	possible	Yes
RDF	explicit	medium	yes	no	No
ZigZag	implicit	medium	yes	no	No
Classical WF	implicit	low	yes	no	No
TRM	**explicit**	**high**	**yes**	**yes**	**Yes**

8.4 Revisiting the Construction Aim

Although the TRM design was done in a context-free environment, fortunately this isolation was not long-lasting and soon the TRM showed its practicality in two ways: It could interconnect the disparate paradigms, and it could be used as a construction kit. The interconnections have been shown to be either known -but not yet formulated- ones like an XML to RDB gateway, or hidden like a Workflow to ZigZag gateway.

Looking at the TRM as a construction kit, i.e. to build new systems directly on top of the TRM as it is, not as restricted by other paradigms, has shown that:

1. By eliminating the associated schema, the TRM was used to develop the TRM-DB, a new schemaless database framework, addressing many flexibility requirements in expressing real-life information (chapter 4).

2. By introducing dynamic ternary links, the TRM was used to build a new general hyperlink model called TRM-NAV, which can be used in different classes of hypertext systems (chapter 5).

3. By introducing TRM-WF, the TRM was shown to be able to define bi-directional and dynamic workflows saving all of the benefits of the classical approaches. As evidence, TWM, a workflow management system based on TRM-WF, was a demonstration of managing simple to complex workflows (chapter 6).

8.5 Other Work Areas

The idea of having a construction kit is a motivation point to find other candidates to be explained in the TRM framework. Also more new link-layer gateways between the known systems can be analyzed or revealed as the future work. The TRM's static and dynamic definition may also be altered or enriched by more elements to be more extendable than the studied works. The areas of such future work are as follows:

1. More Developments on the dynamic side of the TRM: A dynamic TRM-DB, The potential of ZigZag to handle functional links. The Dynamic-TRM may be used to model the link structure of adaptive hypermedia.

2. Building real and practical database management systems on top of the TRM-DB, applying and testing the introduced query method, testing and improvement of the efficiency.

3. Practical development of a generalized hypertext linking engine, to manage the linking structure of a hypertext system. It can be based on top of the Web, or being completely independent.

4. In terms of the workflow technology, this thesis has not provided a full comparison between the TRM-WF based workflow systems and the current commercial workflow products. This comparison and how the TRM can cover and/or help improve each system can be considered as a future work.

5. Some new candidates in computer science can be considered for studying in the TRM framework. If they can be explained by the TRM then it may be a new starting point to discover gateways to connect them to other systems and to analyze or define them in a new way. If they cannot be explained by the TRM, then that may be another starting point to extend the theory of the TRM. Areas like Neural Networks and Route Planning in artificial intelligence have a node-link structure, so they may be considered.

8.6 Why Ternary?

Having 3 elements was shown to be supported by "evidences" in chapter 2. It is now questionable that is this fact just coincidently evidenced by reviewing some related works or is there anything specific to the number of 3?

Recalling the data-information-knowledge pyramid described in chapter 1, it has been shown that moving from data to information is equivalent to discovering the relations between nodes of data (the term "relation" is again used as defined in chapter 1, not in the context of RDBs).

Let us assume that the dimension of the information space is defined as the number of data nodes participating in a relation. In a one-dimensional information space, no relation can be established because the relation is meaningless for a single node.

In the 2-dimensional space, there will be relations to be discovered between couples of nodes. In this case, each node can be either the start or the end of a relation. All one knows about a relation is the "mechanics" of it, i.e. "how" it is established. Moving to the 3-dimensional space, a middle node is added to the two ends for shaping a relation. While two ends could be connected via any middle node, selecting a particular middle node must have an intention or a meaning behind, so the middle node carries some semantics of the relation (or the "why" of it).

In the 3-dimensions case, one may have a far better transition from data to information than the 2-dimensions case, because the relations are richer. So the question will be, can one go to a 4-dimensions information space in order to have an even richer structure? Why not 4 or any other larger number?

What can the fourth element add to our understanding of a relation? Having two middle nodes in a relation does not make any sense because they play the same role (being middle) without any order to be distinguishable. Some other concepts like "weight" or "distance" of a relation can all be merged on to the middle node. The only nodes that cannot be merged on to the middle node are the two ends. Adding the forth node cannot change the nature of a relation and it is observable that the only things that the forth (and upper degree) node can add to the meaning of a relation is an unnecessary complexity[1].

Selecting the 3-dimensional space can be considered as implementing the concept of *Parsimony* in the information modelling. Parsimony is the general scientific tendency to prefer simple solutions over complex ones when choosing between alternative hypothesise [177]. Also when a solution is not enough to choose, it is always preferred to look for a more general one rather a more specific one, as it can increase the resulted predictive power [188].

Parsimony is often associated with *Occam's Razor* which is attributed to William of Ockham (1285–1349). The more accepted quote for Occam's Razor is: *Pluralitas non est ponenda*; or *you must not suppose than more things exist (than you have evidence for)* [41]. Being reductionist is the main message of Occam's Razor, something which was followed in the development of the TRM.

Everything should be made as simple as possible, but not simpler.

This famous quote, which is attributed to Albert Einstein [199], adds a caveat about too much simplicity with the use of Occam's Razor (so it has been dubbed as *Einstein's Razor* [102]). When this is applied to the case of the information space, it is observable that 1-dimension is impossible, 2-dimensions is too simple for most purposes, 3-dimensions is just enough for practical use; and any larger number of dimensions becomes too complex. This is illustrated in Figure 8-3.

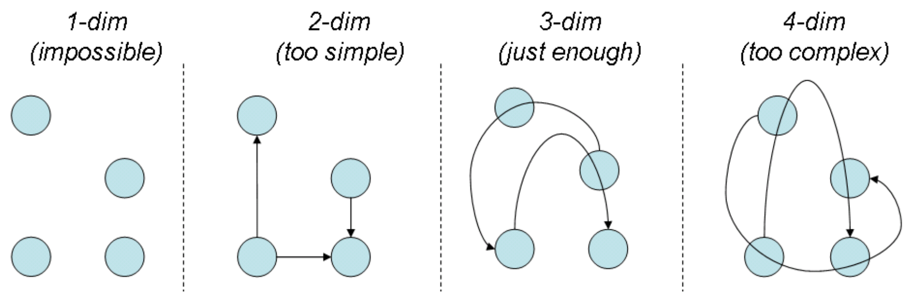


Figure 8-3: Comparing 1, 2, 3 and 4 dimensions in information space – an implementation of Einstein's Razor

[1] Interestingly, although ZigZag was initially designed to be multi-dimensional, it is reducible to a 3-dimensional form (section 0). The multi-dimensional view is an unnecessary complexity in visualization.

Here is where the number of 3 plays its role in a very abstract observation on the topology of a relation. It is possible to consider a very specific role for the number of 3 in the field of information management, for which no other number will suffice.

If this is the case, then the role of 3 can be generalized to *anywhere* that the information plays a role, and consequently conclude that the TRM –or a future variation of it- exists wherever the information does.

8.7 Epilogue

The primary goal of this work has been to find a unified approach to modelling the atoms of information. A proposed solution to this problem is the use of generalized triples of data nodes. This generality provides a range of flexible features – it is simple to define and has no associated schema and properties such as bi-directionality and dynamicity are implicit.

The model that is at the heart of this work – the TRM – is a formal description of such generalized triples in which 1) Each group of three ordered nodes can form a relation; 2) Each relation is itself a node; 3) Relations are reversible – it being possible to traverse them in either direction; and 4) The relations between nodes may be modified dynamically. This approach has been shown to have sufficient flexibility to provide a common underpinning to a number of widely used knowledge-based systems. It may also be used to design implementations of various widely used information paradigms – including hypertext, schemaless databases and workflow management systems.

One of the main values of the TRM is in making the third essential element of the links as explicit as possible. Depending on the application, this value has shown itself in different conceptual layers, from machine readable context (as in TRM-DB) to user's visualization (as in the Ternary links of hypertext). However, it ultimately ensures that the benefits of such a ternary approach are available on the user's level (as in the bi-directional workflows).

The TRM is entirely based upon the number three, and three was shown to be sufficient for the information dimension. Three is an interesting number with special significance in many areas of human endeavour – it seems to be fundamental to human thought. In mathematics it is the first odd prime number, with many other unusual properties. In many aspects of fundamental science, information is grouped into threes (e.g. the "words" of biological encoding in DNA is based upon triplets). Over and over again, philosophical and religious patterns are built out of "three pillars" and the number three has a special significance in many cultures. A provocative possibility is that maybe three is indeed a "magic number" because it plays an important – but hidden – role in the very fabric of information.

There have been some suggestions found in the development of the TRM that it is likely to be extensible to other levels of the knowledge-hierarchy. Knowledge-oriented hypermedia (section 2.4.1) supports the utilization of the TRM to model the knowledge representation in hypermedia systems, and TRM-WF (chapter 6) is capable of supporting the decision making process in workflows. An interesting and ambitious view is that maybe the TRM truly is a unified theory of information. Maybe in the future it might be possible to extend the initial problem considering the atoms of information to look at atoms of knowledge, or maybe even atoms of wisdom.

Chapter 9-

BIBLIOGRAPHY

[1] W. M. P. v. d. Aalst, "The Application of Petri Nets to Workflow Management", *The Journal of Circuits, Systems and Computers*, vol. 8, pp. 21-66, 1998.

[2] W. M. P. v. d. Aalst, L. Aldred, M. Dumas, and A. H. M. t. Hofstede, "Design and Implementation of the Yawl System", *Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAISE' 04), Riga, Latvia*, 2004.

[3] W. M. P. v. d. Aalst and K. v. Hee, *Workflow Management*: MIT Press, 2004.

[4] W. M. P. v. d. Aalst and A. H. M. t. Hofstede, "Yawl: Yet Another Workflow Language", *QUT Technical Report, FIT-TR-2002-06, Queensland University of Technology, Brisbane*, 2002.

[5] W. M. P. v. d. Aalst and A. H. M. t. Hofstede, "Yawl: Yet Another Workflow Language (Revised Version)", *QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane*, 2003.

[6] W. M. P. v. d. Aalst, A. H. M. t. Hofstede, B. Kiepuszewski, and A. P. Barros, "Advanced Workflow Patterns", the 7th International Conference on Cooperative Information Systems (CoopIS 2000), 2000.

[7] W. M. P. V. D. Aalst, A. H. M. T. Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow Patterns", *Distributed and Parallel Databases*, vol. 14, pp. 5-51, 2003.

[8] W. M. P. v. d. Aalst, H. M. W. Verbeek, and A. Kumar, "Xrl/Woflan: Verification of an Xml/Petri-Net Basedlanguage for Inter-Organizational Workflows", in *Proceedings of the 6th Informs Conference on Information Systems and Technology (CIST-2001)*, 2001, pp. 30-45.

[9] K. R. Abbott and S. K. Sarin, "Experiences with Workflow Management: Issues for the Next Generation", in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. Chapel Hill, North Carolina, United States: ACM Press, 1994.

[10] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and Xml*: Morgan Kaufmann Publishers Inc., 2000.

[11] R. L. Ackoff, "From Data to Wisdom", *Journal of Applied Systems Analysis*, vol. 16, pp. 3-9, 1981.

[12] J. Allan, "Automatic Hypertext Link Typing", Proceedings of the seventh ACM conference on Hypertext, Bethesda, USA, 1996.

[13] G. Alonso, D. Agrawal, A. E. Abbadi, and C. Mohan, "Functionality and Limitations of Current Workflow Management Systems ", *Submitted to IEEE Expert*, 1997.

[14] G. Antoniou and F. van-Harmelen, *A Semantic Web Primer*: the MIT Press, 2004.

[15] F. Arciniegas. "What Is Xlink?" Retrieved 15/08/2005, from http://www.xml.com/lpt/a/2000/09/xlink/index.html, 2000.

[16] H. C. Arents and W. F. L. Bogaerts, "Information Structuring for Intelligent Hypermedia: A Knowledge Engineering Approach", Proceedings of the 3rd International Conference on Database and Expert Systems Applications, Valencia, 1992.

[17] H. C. Arents and W. F. L. Bogaerts, "Towards an Architecture for Third-Order Hypermedia Systems", *Hypermedia*, vol. 3, pp. 133-152, 1991.

[18] AristaFlow. "Next Generation Enterprise Process Management: Component-Oriented Development of Adaptive Process-Oriented Enterprise Software." Retrieved 10/2006, from http://www.aristaflow.de, 2006.

[19] H. Ashman, "Relations Modelling Sets of Hypermedia Links and Navigation", *The Computer Journal*, vol. 43, pp. 345-363, 2000.

[20] H. Ashman, "Theory and Practice of Large-Scale Hypermedia Systems": PhD Thesis, The Royal Melbourne Institute of Technology, Australia, 1997.

[21] H. Ashman, "What Is Hypermedia?" *SIGWEB Newsletter*, vol. 3, pp. 6-8, 1994.

[22] H. Ashman, A. Garrido, and H. O. Kukkonen, "Hand-Made and Computed Links, Precomputed and Dynamic Links", Proceedings of Hypermedia - Information Retrieval - Multimedia '97 (HIM '97), Dortmund, Germany, 1997.

[23] R. M. Baecker, D. Nastos, I. R. Posner, and K. L. Mawby, "The User-Centered Iterative Design of Collaborative Writing Software", in *Proceedings of the SIGCHI conference on Human factors in computing systems*. Amsterdam, The Netherlands: ACM Press, 1993.

[24] T. Berners-Lee. "Notation 3 (N3)." Retrieved June 2006, from http://www.w3.org/DesignIssues/Notation3, 1998.

[25] T. Berners-Lee, "Semantic Web Road Map", *W3C Electronic Journal*, vol. Sep. 1998, 1998.

[26] T. Berners-Lee, *Weaving the Web, by Its Inventor*: Texere LLC, 2000.

[27] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, vol. 284, pp. 34-44, 2001.

[28] A. Bhaumik, D. Dixit, R. Galnares, A. Krishna, M. Tzagarakis, M. Vaitis, M. Bieber, V. Oria, Q. Lu, F. Alljalad, and L. Zhan, "Towards Hypermedia Support for Database Systems", Proceedings of the 34th Hawaiian International Conference on System Sciences, Hawaii, 2001.

[29] M. Bieber. "Dynamic Hypermedia Engine (D.H.E)." CIS Department, New Jersey Institute of Technology, Retrieved 15/08/2005, from http://www.cis.njit.edu/~bieber/dhe-overview.html, 2000.

[30] M. Bieber, "Hypertext and Web Engineering", Proceedings of the ninth ACM conference on Hypertext and hypermedia: links, objects, time and space-structure in hypermedia systems, Pittsburgh, USA, 1998.

[31] M. Bieber, R. Galnares, and Q. Lu, "Web Engineering and Flexible Hypermedia", Proceedings of the 2nd workshop on adaptive hypertext and hypermedia (HT' 98), Pittsburgh, USA, 1998.

[32] M. Bieber, F. Vitali, H. Ashman, V. Balasubramanian, and H. Oinas-Kukkonen, "Fourth Generation Hypermedia: Some Missing Links for the World Wide Web", *International Journal of Human-Computer Studies*, vol. 47, pp. 31-65, 1997.

[33] A. Borgida. "Storing X.M.L in a R.D.B.M.S." Lecture Notes, Department of Computer Science, Rutgers University, Retrieved 15/08/2005, from http://www.cs.rutgers.edu/~borgida/336/xml2db4.pdf, 2003.

[34] P. D. Bra, G.-J. Houben, and H. Wu, "A.H.A.M: A Dexter-Based Reference Model for Adaptive Hypermedia", in *Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots*. Darmstadt, Germany: ACM, 1999.

[35] M. Brambilla, "Extending Hypertext Conceptual Models with Process-Oriented Primitives", in *Lecture Notes in Computer Science*, vol. 2813: Springer Berlin/Heidelberg, 2003, pp. 246-262.

[36] M. Brambilla, "Generation of Webml Web Application Models from Business Process Specifications", in *Proceedings of the 6th international conference on Web engineering*. Palo Alto, California, USA: ACM, 2006.

[37] P. Brusilovsky, "Adaptive Hypermedia ", *User Modeling and User-Adapted Interaction*, vol. 11, pp. 87-110, 2001.

[38] P. Brusilovsky, "The Adaptive Web", *LNCS*, vol. 4321, pp. 263-290, 2007.

[39] P. Brusilovsky, "Methods and Techniques of Adaptive Hypermedia", *User Modeling and User-Adapted Interaction*, vol. 6, pp. 87-129, 1996.

[40] S. Buckingham-Shum, "The Missing Link: Hypermedia Usability Research & the Web", *SIGCHI Bulletin*, vol. 28, 1996.

[41] C. D. Burns, "Occam's Razor", *Mind, New Series*, vol. 24, pp. 592, 1915.

[42] C. Bussler, "Enterprise-Wide Workflow Management", *IEEE Concurrency*, vol. 7, pp. 32-43, 1999.

[43] J. P. Carlisle, "A Look into the Relationship between Knowledge Management and the Knowledge Hierarchies ", *Proceeding of the 40th Annual Hawaii International Conference on System Sciences*, pp. 183a, 2007.

[44] L. Carr, W. Hall, S. Bechhofer, and C. Goble, "Conceptual Linking: Ontology-Based Open Hypermedia", Proceedings of the 10th international conference on World Wide Web, Hong Kong, 2001.

[45] L. Carr, D. D. Roure, W. Hall, and G. Hill, "The Distributed Link Service: A Tool for Publishers, Authors and Readers", Proceedings of the 4th International WWW Conference, Boston, USA, 1995.

[46] S. Ceri, P. Fraternali, and A. Bongio, "Web Modeling Language (Web M.L.): A Modeling Language for Designing Web Sites", *Computer Networks*, vol. 33, pp. 137-157, 2000.

[47] P. E. Ceruzzi, *A History of Modern Computing*: MIT Press, 2003.

[48] D. Chamberlin, "Xquery: A Query Language for Xml", in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. San Diego, California: ACM, 2003.

[49] G. Chartrand, *Introductory Graph Theory*. New York: Dover, 1985.

[50] Y. M. S. Chen S.M., "Generating Fuzzy Rules from Relational Database Systems for Estimating Null Values ", *Cybernetics and Systems*, vol. 28, pp. 695-723, 1997.

[51] X. Chen, D.-H. Kim, N. Nnadi, H. Shah, P. Shrivastava, M. Bieber, I. Im, and Y.-F. Wu. "Integrating Web Systems through Linking." www2003.org, Retrieved 15/08/2005, from http://www2003.org/cdrom/papers/poster/p145/p145-chen.htm, 2004.

[52] S. Choenni, R. Bakker, and W. Baets, "On the Evaluation of Workflow Systems in Business Processes", *Electronic Journal of Information Systems Evaluation*, 2003.

[53] N. Chomsky, "Three Models for the Description of Language", *IEEE Transactions on Information Theory*, vol. 2, pp. 113- 124, 1956.

[54] S. P. Christodoulou, G. D. Styliaras, and T. S. Papatheodrou, "Evaluation of Hypermedia Application Development and Management Systems", Proceedings of the ninth ACM conference on hypertext and hypermedia: links, objects, time and space-structure in hypermedia systems, Pittsburgh, USA., 1998.

[55] E. F. Codd, "Extending the Database Relational Model to Capture More Meaning", *ACM Transaction on Database Systems*, vol. 4, pp. 397-434, 1979.

[56] E. F. Codd, "Further Normalization of the Data Base Relational Model", *IBM Research Report, San Jose, California*, vol. RJ909, 1971.

[57] E. F. Codd, "Relational Database: A Practical Foundation for Productivity", *Communications of ACM*, vol. 25, pp. 109-117, 1982.

[58] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of ACM*, vol. 13, pp. 377-387, 1970.

[59] G. H. Collier, "Thoth-Ii: Hypertext with Explicit Semantics", in *Proceeding of the ACM conference on Hypertext*. Chapel Hill, USA: ACM Press, 1987.

[60] J. H. D. Helic, H. Maurer, "An Analysis of Application of Business Process Management Technology in E-Learning Systems, Technical Report", Institue for Information Systems and Computer Media, Graz University of Technology, Austria 2005.

[61] H. Darwen. "How to Handle Missing Information without Using Nulls, Presentation in Warwick University." Retrieved 01/02/2009, from http://www.dcs.warwick.ac.uk/~hugh/TTM/Missing-info-without-nulls.pdf, 2006.

[62] C. J. Date and H. Darwen, *Databases, Types, and the Relational Model: The Third Manifesto*: Addison-Wesley, 2006.

[63] H. Davis, W. Hall, I. Heath, G. Hill, and R. Wilkins, "Towards an Integrated Information Environment with Open Hypermedia Systems", in *Proceedings of the ACM conference on Hypertext*. Milan, Italy: ACM, 1992.

[64] H. C. Davis, S. Reich, and D. E. Millard, "A Proposal for a Common Navigational Hypertext Protocol", Technical Report, Department of Electronic and Computer Science, The University of Southampton 1997.

[65] DBIS. "Adept - Next Generation Workflow Technology." Retrieved 10/2006, from http://www.informatik.uni-ulm.de/dbis/, 2006.

[66] D. Degler, S. Henninger, and L. Battle, "Semantic Web Hci: Discussing Research Implications", in *CHI '07 extended abstracts on Human factors in computing systems*. San Jose, CA, USA: ACM, 2007.

[67] J. Desel and G. Juhas, "What Is Petri Net? Informal Answers for the Informed Reader", in *Lecture Notes in Computer Science*, vol. 2128, H. Ehrig, Ed. Berlin: Springer-Verlag, 2001, pp. 1-25.

[68] M. H. Diimitrios Georgakopoulos, Amit Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure ", *Distributed and Parallel Databases*, vol. 3, pp. 119-153, 1995.

[69] J. Domingue and M. Dzbor, "Magpie: Supporting Browsing and Navigation on the Semantic Web", in *Proceedings of the 9th international conference on Intelligent user interfaces*. Funchal, Madeira, Portugal: ACM, 2004.

[70] M. Dumas and A. t. Hofstede, "Uml Activity Diagrams as a Workflow Specification Language", *Proceedings of the International Conference on the Unified Modeling Language (UML), Toronto, Canada*, 2001.

[71] C. H. L. Eric W.T. Ngai, Y.C. Wong, "Application of the Workflow Management System in Electronic Commerce", *International Journal of Business Information Systems*, vol. 1, pp. 182-198, 2005.

[72] R. Z. N. F Harary, D Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs*: John Wiley & Sons, 1965.

[73] F. M. Facca and M. Brambilla, "Extending Web.M.L. Towards Semantic Web", in *Proceedings of the 16th international conference on World Wide Web*. Banff, Alberta, Canada: ACM, 2007.

[74] B. Fallenstein and T. J. Lukka. "Hyperstructure: Computers Build around Things That We Care About." Retrieved 15/08/2005, from http://fenfire.org/manuscripts/2004/hyperstructure/, 2004.

[75] J. B. P. L. Faucher, A. M. Everett, and R. Lawson, "Reconstituting Knowledge Management", *Journal of Knowledge Management (in press)*, 2008.

[76] G. Fischer, R. McCall, and A. Morch, "Janus: Integrating Hypertext with a Knowledge-Based Design Environment", Proceedings of the second annual ACM conference on Hypertext, Pittsburgh, USA., 1989.

[77] D. Florescu and D. Kossmann, "A Performance Evaluation of Alternative Mapping Schemes for Storing Xml Data in a Relational Database", *Inria Report of Research*, 1999.

[78] D. Florescu and D. Kossmann, "Storing and Querying Xml Data Using an Rdmbs", *Bulletin of the IEEE Computer Society, Technical Committee on Data Engineering*, 1999.

[79] A. M. Fountain, W. Hall, I. Heath, and H. C. Davis, "Microcosm: An Open Model for Hypermedia with Dynamic Linking", in *Hypertext: Concepts, Systems and Applications*: Cambridge University Press, 1992, pp. 298-311.

[80] H. P. Frei and D. Stieger, "Making Use of Hypertext Links When Retrieving Information", Proceedings of the ACM conference on Hypertext, Milan, Italy, 1992.

[81] R. Furuta and P. D. Stotts, "Programmable Browsing Semantics in Trellis", Proceedings of the second annual ACM conference on Hypertext, Pittsburgh, USA., 1989.

[82] A. Garnemark, "Workflow and Knowledge Management": Master Thesis, School of Economics and Commercial Law, Department of Informatics, University of Goteborg, 2002.

[83] A. Garnemark, "Workflow and Knowledge Management (Msc Thesis)", in *University of Goteborg, Department of Informatics*, 2002, pp. 31-43.

[84] J. Gaulding and B. Katz, "Using "Word-Knowledge" Reasoning for Question Answering", in *The Society of Text: Hypertext, Hypermedia, and the Social Construction of Information*: MIT Press, 1989, pp. 403-422.

[85] D. C. Gene Bellinger, Anthony Mills. "Data, Information, Knowledge, and Wisdom." Retrieved July 2008, from http://www.systems-thinking.org/dikw/dikw.htm, 2004.

[86] D. Georgakopoulos, M. Hornick, and A. Sheth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure ", *Distributed and Parallel Databases*, vol. 3, pp. 119-153, 1995.

[87] N. Gibbins, Harris, S., Michaelides, D. T., Millard, D. E., Weal, M. J., "Exploring the Relationship between Fohm and Rdf", 1st International Workshop on Hypermedia and the Semantic Web, Nottingham, UK, 2003.

[88] C. Goble and S. Bechhofer, "The Return of the Prodigal Web", Proceedings of the 18th ACM Conference of Hypertext and Hypermedia (HT'07), Manchester, UK, 2007.

[89] C. Goble, S. Bechhofer, L. Carr, D. D. Roure, and W. Hall, "Conceptual Open Hypermedia = the Semantic Web? " The Second International Workshop on the Semantic Web, Hong Kong, 2001.

[90] G. Grahne, "Dependency Satisfaction in Databases with Incomplete Information", in *Proceedings of the 10th International Conference on Very Large Data Bases*: Morgan Kaufmann Publishers Inc., 1984.

[91] V. A. Gruzman and V. I. Senichkin, "Hypermedia Models", *Automated Remote Control*, vol. 62, pp. 677-694, 2001.

[92] F. Halasz and M. Schwartz, "The Dexter Hypertext Reference Model", *Communications of the ACM*, vol. 37, pp. 30-39, 1994.

[93] W. Hall, G. Hill, and H. Davis, "The Microcosm Link Service", Proceedings of the fifth ACM conference on Hypertext, Seattle, USA, 1993.

[94] C. H. P. Harry R. Lewis, *Elements of the Theory of Computation*, 2nd ed: Prentice-Hall, 1998.

[95] S. Hawking, *A Brief History of Time*: Bantam Books, 1988.

[96] S. W. Hawking, *The Illustrated Theory of Everything: The Origin and Fate of the Universe*: New Millennium Press, 2004.

[97] J. Hey. "The Data, Information, Knowledge, Wisdom Chain: The Metaphorical Link." Intergovernmental Oceanographic Commission, Retrieved August 2008, from http://best.me.berkeley.edu/~jhey03/files/reports/IS290_Finalpaper_HEY.pdf, 2004.

[98] D. R. Hofstadter, *Godel, Escher, Bach: An Eternal Golden Braid*: Basic Books, Inc., 1979.

[99] W. P. Initiative. "Workflow Patterns." Retrieved 7/2007, from http://www.workflowpatterns.com, 2003.

[100] T. Isakowitz, E. A. Stohr, and P. Balasubramanian, "Rmm: A Methodology for Structured Hypermedia Design", *Communications of the ACM*, vol. 38, pp. 34-44, 1995.

[101] T. B. J. Bosak, "Xml and the Second-Generation Web ", *Scientific American*, vol. 280, pp. 89, 1999.

[102] Q. T. Jackson. "On Einstein's Razor: Telesis-Driven Introduction of Complexity into Apparently Sufficiently Non-Complex Linguistic Systems." Submitted to Progress in Complexity, Information, and Design, Retrieved August 2008, from http://www.thothic.com/downloads/Jackson_EinsteinsRazor_050205.pdf, 2005.

[103] R. James, J. Ivar, and B. Grady, *The Unified Modeling Language Reference Manual*: Addison-Wesley Longman Ltd., 1999.

[104] B. Kemme. "Workflow Management Systems (Lecture Notes)." Retrieved 7/2004, from http://www.cs.mcgill.ca/~kemme/cs764/lectures/764-workflow.pdf, 2000.

[105] M. Klein, J. Broekstra, D. Fensel, F. v. Harmelen, and I. Horrocks, "Ontologies and Schema Languages on the Web", in *Spinning the Semantic Web*, J. H. Dieter Fensel, Henry Lieberman, Wolfgang Wahlster, Ed.: MIT Press, 2003, pp. 95-139.

[106] P. H. Lewis, W. Hall, L. A. Carr, and D. D. Roure, "The Significance of Linking", *ACM Computer Surveys*, vol. 31, pp. 10, 1999.

[107] F. Leymann and D. Roller. "Workflow-Based Applications." Retrieved 12/2005, from http://www.research.ibm.com/journal/sj/361/leymann.html, 1997.

[108] Y. E. Lien, "Multivalued Dependencies with Null Values in Relational Databases", *Proceedings of the 5th International Conference on Very Large Databases*, pp. 155-168, 1971.

[109] M. Lousa, A. Sarmento, and A. Machado, "Expectations Towards the Adoption of Workflow Systems: The Results of a Case Study", in *Proceedings of the 6th International Workshop on Groupware*: IEEE Computer Society, 2000.

[110] T. J. Lukka and K. Ervasti. "Gzigzag - a Platform for Cybertext Experiments." Retrieved 15/08/2005, from http://www.nongnu.org/gzz/ct/ct-ns4.html, 2002.

[111] S. C. M. Brambilla, S. Comai, P. Fraternali, I. Manolescu, "Specification and Design of Workflow-Driven Hypertexts", *Journal of Web Engineering*, vol. 1, 2003.

[112] E. Mahmoud Omar Eliwa, "Benefits and Limitations of the Semantic Web": Msc Thesis, The University of Nottingham, UK, 2003.

[113] D. Maier, *The Theory of Relational Databases*: Computer Science Press, 1983.

[114] P. Mamaani and S. Abdul Kareem, "Uml Extensions for Hypermedia Navigation and Presentation Modelling", ACM Symposium on Software Visualization (SOFTVIS '05), St. Louis, Missouri, USA, 2005.

[115] C. C. Marshall and F. M. Shipman, "Which Semantic Web?" in *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*. Nottingham, UK: ACM Press, 2003, pp. 57-66.

[116] M. J. McGuffin. "A Graph-Theoretic Introduction to Ted Nelson's Zzstructures." Retrieved Oct. 2004, from http://www.dgp.toronto.edu/~mjmcguff/research/zigzag/, 2004.

[117] M. J. McGuffin and M. C. Schraefel, "A Comparison of Hyperstructures: Zzstructures, Mspaces, and Polyarchies", Proceedings of the fifteenth ACM conference on Hypertext & hypermedia, Santa Cruz, CA, USA, 2004.

[118] A. Mili, "A Relational Approach to the Design of Deterministic Programs", *Acta Informatica*, vol. 20, pp. 315-328, 1983.

[119] D. E. Millard, L. Moreau, H. C. Davis, and S. Reich, "Fohm: A Fundamental Open Hypertext Model for Investigating Interoperability between Hypertext Domains", Proceedings of the eleventh ACM conference on Hypertext and hypermedia, San Antonio, USA, 2000.

[120] M. F. Mohageg, "The Influence of Hypertext Linking Structures on the Efficiency of Information Retrieval", *Human Factors*, vol. 34, pp. 351-367, 1992.

[121] A. Moore and T. J. Brailsford, "Unified Hyperstructures for Bioinformatics: Escaping the Application Prison", *Journal of Digital Information*, vol. 5, 2004.

[122] A. Moore, J. Goulding, T. Brailsford, and H. Ashman, "Practical Applitudes: Case Studies of Applications of the Zigzag Hypermedia System", Proceedings of the fifteenth ACM conference on Hypertext and hypermedia, Santa Cruz, CA, USA, 2004.

[123] L. Moreau and W. Hall, "On the Expressiveness of Links in Hypertext Systems", *The Computer Journal*, vol. 41, pp. 459-473, 1998.

[124] S. S. Mysore Ramaswamy, Ye-Sho Chen, "Using Directed Hypergraphs to Verify Rule-Based Expert Systems", *IEEE Transaction on Knowledge and Data Engineering*, vol. 9, pp. 221-237, 1997.

[125] J. Nanard and M. Nanard, "Using Structured Types to Incorporate Knowledge in Hypertext", Proceedings of the third annual ACM conference on Hypertext, San Antonio, USA, 1991.

[126] J. Nanard, M. Nanard, and H. Richy, "Conceptual Documents: A Mechanism for Specifying Active Views in Hypertext", Proceedings of the ACM conference on document processing systems, Santa Fe, USA, 1988.

[127] M. Nanard, J. Nanard, and P. King, "Iuhm: A Hypermedia-Based Model for Integrating Open Services, Data and Metadata", Proceedings of the fourteenth ACM conference on hypertext and hypermedia, Nottingham, UK, 2003.

[128] T. H. Nelson, "Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate", Proceedings of ACM national conference, Cleveland, USA, 1965.

[129] T. H. Nelson, "A Cosmology for a Different Computer Universe: Data Model, Mechanisms, Virtual Machine and Visualization Infrastructure", *Journal of Digital Information*, vol. 5, 2004.

[130] T. H. Nelson, *Literary Machines*: Swarthmore, PA, 1981.

[131] T. H. Nelson. "What's on My Mind." Retrieved August 2008, from http://www.xanadu.com.au/ted/zigzag/xybrap.html, 1998.

[132] T. H. Nelson, "Zigzag (Tech Briefing)", in *Proceedings of the 12th ACM conference on Hypertext and Hypermedia*. Rhus, Denmark: ACM, 2001.

[133] J. Noll and W. Scacchi, "Specifying Process-Oriented Hypertext for Organizational Computing", *Journal of Networking and Computer Applications*, vol. 24, pp. 39-61, 2001.

[134] P. J. Nürnberg and H. Ashman, "What Was the Question? Reconciling Open Hypermedia and World Wide Web Research", Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots, Darmstadt, Germany, 1999.

[135] P. J. Nürnberg, J. J. Leggett, and E. R. Schneider, "As We Should Have Thought", Proceedings of the eighth ACM conference on Hypertext, Southampton, UK, 1997.

[136] P. J. Nürnberg, U. K. Wiil, and D. L. Hicks, "Rethinking Structural Computing Infrastructures", Proceedings of the fifteenth ACM conference on Hypertext and hypermedia, Santa Cruz, CA, 2004.

[137] K. Nyberg. "Workflow Definition Languages (Seminar on Database Management)." Retrieved 12/2005, from http://www.cs.hut.fi/~kny/workflowlang/, 2000.

[138] Object-Management-Group, "Business Process Modeling Notation (Bpmn) Specification, Final Adopted Specification", DTC/06-02-01, 2006.

[139] Object-Management-Group. "Unified Modeling Language: Superstructure, Version 2.1.1 (with Change Bars)." Retrieved July 2007, from http://www.omg.org/docs/formal/07-02-03.pdf, 2007.

[140] H. Oinas-Kukkonen, "Embedding Hypermedia into Information Systems", in *Proceedings of the 30th Hawaiian International Conference on System Sciences: Digital Documents - Volume 6*: IEEE Computer Society, 1997, pp. 187.

[141] H. Oinas-Kukkonen, "What Is inside a Link?" *Communications of the ACM*, vol. 41, pp. 98, 1998.

[142] S. B. Palmer. "The Semantic Web: An Introduction." Retrieved 15/08/2005, from http://infomesh.net/2001/swintro/, 2001.

[143] F. Pascal, *Practical Issues in Database Management: A Reference for the Thinking Practitioner*: Addison-Wesley, 2000.

[144] A. Pearl, "Sun's Link Service: A Protocol for Open Linking", in *Proceedings of the second annual ACM conference on Hypertext*. Pittsburgh, Pennsylvania, United States: ACM, 1989.

[145] J. L. Peterson, "Petri Nets", *ACM Computer Survey*, vol. 9, pp. 223-252, 1977.

[146] C. A. Petri, "Kommunikation Mit Automaten, Phd Thesis", *Rheinisch-Westfälisches Institut fur instrumentelle Mathematik*, 1962.

[147] G. Pettrash, "Managing Knowledge Assets for Value", *Proceedings of the Knowledge-Based Leadership Conference*, 1996.

[148] H. Podeswa, *Business Object-Oriented Modeling for Business Analysts*: Course Technology Incorporated, 2005.

[149] A. Pourabdollah, "A User-Friendly Process Description Language Used in Creating Database Model of Workflows": M.Sc. Thesis, The University of Nottingham, Malaysia Campus., 2004.

[150] A. Pourabdollah, H. Ashman, and T. Brailsford, "Are We Talking About the Same Structure? A Unified Approach to Hypertext Links, Xml, Rdf and Zigzag", Proceedings of the nineteenth ACM conference on Hypertext and hypermedia, Pittsburgh, PA, USA, 2008.

[151] A. Pourabdollah, T. Brailsford, and H. Ashman, "A User-Oriented Design for Business Workflow Systems", in *Lecture Notes in Computer Science - Teaa' 2006*, vol. 4473, D. Draheim and G. Weber, Eds. Berlin Heidleberg: Springer-Verlag, 2007, pp. 285-297.

[152] A. Pourabdollah and M. Hartley, "Gathering Unstructured Workflow Data into Relational Database Model Using Process Definition Language", in *Proceedings of the 24th IASTED international conference on Database and applications*. Innsbruck, Austria: ACTA Press, 2006.

[153] Z. Qiu, "Hyperstructure-Based Search Methods for the World Wide Web": PhD Thesis, Darmstadt Technical University, Denmark, 2004.

[154] W. C. Recommendation. "Rdf Test Cases." Retrieved June 2006, from http://www.w3.org/TR/rdf-testcases/, 2004.

[155] S. Reich, U. K. Wiil, P. J. Nurnberg, H. C. Davis, K. Graonbaek, K. M. Anderson, D. E. Millard, and J. M. Haake, "Addressing Interoperability in Open Hypermedia: The Design of the Open Hypermedia Protocol ", *New Rev Hypermedia Multimedia*, vol. 5, pp. 207-248, 1999.

[156] H. A. Reijers, *Design and Control of Workflow Processes: Business Process Management for the Service Industry*: Springer-Verlag New York, Inc., 2003.

[157] J. Rowley, "The Wisdom Hierarchy: Representations of the Dikw Hierarchy", *Journal of Information Science*, vol. 33, pp. 163-180, 2007.

[158] W. W. Royce, "Managing the Development of Large Software Systems", *Proceedings, IEEE WESCON*, pp. 1-9, 1970.

[159] N. Russell, W. M. P. v. d. Aalst, and A. H. M. t. Hofstede, "Exception Handling Patterns in Process-Aware Information Systems", *BPM Center Report BPM-06-04, BPMcenter.org*, 2006.

[160] N. Russell, A. H. M. t. Hofstede, W. M. P. v. d. Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View", *BPM Center Report BPM-06-22, BPMcenter.org*, 2006.

[161] N. Russell, A. H. M. t. Hofstede, D. Edmond, and W. M. P. v. d. Aalst, "Workflow Data Patterns", *QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane*, 2004.

[162] N. Russell, A. H. M. t. Hofstede, D. Edmond, and W. M. P. v. d. Aalst, "Workflow Resource Patterns", *BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven*, 2004.

[163] A. Sarmento and A. Machado, "Impact Evaluation of Organisational Changes Enabled by Workflow Systems ", *Proceedings of the 6th International Workshop on Groupware (CRIWG'00)*, pp. 134, 2000.

[164] T. Schael and B. Zeller, "Workflow Management Systems for Financial Services", in *Proceedings of the conference on Organizational computing systems*. Milpitas, California, United States: ACM, 1993.

[165] M. C. Schraefel, J. Golbeck, D. Degler, A. Bernstein, and L. Rutledge, "Semantic Web User Interactions: Exploring Hci Challenges", in *HCI '08 extended abstracts on Human factors in computing systems*. Florence, Italy: ACM, 2008.

[166] N. Sharma. "The Origin of the Data Information Knowledge Wisdom Hierarchy." Retrieved July 2008, from http://www-personal.si.umich.edu/~nsharma/dikw_origin.htm, 2004.

[167] F. M. Shipman and C. C. Marshall, "Spatial Hypertext: An Alternative to Navigational and Semantic Links", *ACM Computer Surveys*, vol. 31, pp. 14, 1999.

[168] B. Signer and M. Norrie, "As We May Link: A General Metamodel for Hypermedia Systems ", in *Lecture Notes in Computer Science*, vol. 481: Springer Berlin/Heidelberg, 2008, pp. 359-374.

[169] P. D. Stotts and R. Furuta, "Adding Browsing Semantics to the Hypertext Model", Proceedings of the ACM conference on document processing systems, Santa Fe, USA, 1988.

[170] P. D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics", *ACM Transactions on Information Systems*, vol. 7, pp. 3-29, 1989.

[171] D. Suciu, "Semistructured Data and Xml", in *Information Organization and Databases: Foundations of Data Organization*: Kluwer Academic Publishers, 2000, pp. 9-30.

[172] K. Takahashi, "Metalevel Links: More Power to Your Links", *Communications of the ACM*, vol. 41, pp. 103-105, 1998.

[173] P. Thistlewaite, "Automatic Construction and Management of Large Open Webs", *Information Processing and Management*, vol. 33, pp. 161-173, 1997.

[174] L. P. Thomas H Davenport, *Working Knowledge, How Organizations Manage What They Know*: Harvard Business School Press, Boston, MA, 1998.

[175] F. Tian, D. J. DeWitt, J. Chen, and C. Zhang, "The Design and Performance Evaluation of Alternative Xml Storage Strategies", *SIGMOD Rec.*, vol. 31, pp. 5-10, 2002.

[176] A. M. Turing, "Computing Machinery and Intelligence ", *Mind*, vol. 59, pp. 433-460, 1950.

[177] D. S. Vaknin. "Parsimony – the Fourth Substance." Retrieved August 2008, from http://samvak.tripod.com/parsimony.html.

[178] Y. Vassiliou, "Null Values in Data Base Management a Denotational Semantics Approach", in *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*. Boston, Massachusetts: ACM, 1979.

[179] J. Verbyla and H. Ashman, "A User-Configurable Hypermedia-Based Interface Via the Functional Model of the Link", *Hypermedia*, vol. 6, pp. 193-208, 1994.

[180] S. M. Vivekanandan and D. C. De Roure, "Workflow Description for Open Hypermedia Systems", *Proceedings of the International Workshop on Open Hypermedia Systems Core Concepts and Research Directions, Pre-Conference Workshop at the ACM 13th International Conference on Hypertext and Hypermedia (HT'02)*, pp. 52-56, 2002.

[181] W3C-Working-Group. "Defining N-Ary Relations on the Semantic Web." Retrieved June 2006, from http://www.w3.org/TR/swbp-n-aryRelations/, 2006.

[182] W3C. "Html 3.2 Reference Specification." Retrieved August 2008, from http://www.w3.org/TR/REC-html32-19970114, 1999.

[183] W3C. "Html 4.01 Specification." Retrieved August 2008, from http://www.w3.org/TR/REC-html40/, 1999.

[184] W3C. "Resource Description Framework (Rdf): Concepts and Abstract Syntax." Retrieved Feb 2004, from http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2006.

[185] W3C. "Xml Linking Language (Xlink) Version 1.0." Retrieved 17/07/2008, from http://www.w3.org/TR/xlink/, 2001.

[186] A. Warren. "Microcosm: An Open Hypermedia System." University of Southampton, Retrieved 15/08/2005, from http://www.soton.ac.uk/~connect/misc/mcosm1.html#http://www.soton.ac.uk/~connect/misc/mcosm1.html#, 1995.

[187] A. Warren. "Microcosm: Flexibly Linking Information." University of Southampton, Retrieved 15/08/2005, from http://www.soton.ac.uk/~connect/v6i1/mcosm2.html#http://www.soton.ac.uk/~connect/v6i1/mcosm2.html#, 1995.

[188] G. I. Webb, "Generality Is More Significiant Than Complexity, Towards an Alternative to Occam's Razor", *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence*, pp. 60-67, 1994.

[189] B. P. M. I. Website. Retrieved July 2007, from http://www.bpmi.org.

[190] B. P. M. N. Website. Retrieved July 2007, from http://www.bpmn.org.

[191] O. M. G. Website. Retrieved July 2007, from http://www.omg.org.

[192] Y. A. W. L. Website. "Yawl: Yet Another Workflow Language." Retrieved 7/2007, from http://yawlfoundation.org/, 2007.

[193] H. Weinreich, H. Obendorf, and W. Lamersdorf, "The Look of the Link - Concepts for the User Interface of Extended Hyperlinks", Proceedings of the twelfth ACM conference on Hypertext and Hypermedia, Århus, Denmark, 2001.

[194] M. Weske, T. Goesmann, R. Holten, and R. Striemer, "A Reference Model for Workflow Application Development Processes", *SIGSOFT Software Engineering Notes*, vol. 24, pp. 1-10, 1999.

[195] WFMC. "The Workflow-Management-Coalition." Retrieved 8/2006, from http://www.wfmc.org/, 2006.

[196] WFMC, "Workflow Process Definition Interface -- Xml Process Definition Language", Workflow Management Coalition WFMC-TC-1025, October 25 2002.

[197] S. A. White, "Introduction to Bpmn", *BPTrends*, July 2004.

[198] S. A. White. "Process Modeling Notations and Workflow Patterns." Retrieved 12/2005, from http://www.omg.org/bp-corner/bp-files/Process_Modeling_Notations.pdf, 2004.

[199] Wikiquote. "Albert Einstein." Retrieved August 2008, from http://en.wikiquote.org/wiki/Albert_Einstein.

[200] K. Wilber, *A Theory of Everything*. Boston: Shambhala Publications, 2001.

[201] Workflow-Patterns-Initiative. "Workflow Patterns." Retrieved 7/2007, from http://www.workflowpatterns.com, 2007.

[202] C. Zaniolo, "Database Relations with Null Values", in *Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*. Los Angeles, California: ACM, 1982.

[203] L. Zhang and M. Bieber, "Towards Just-in-Time Hypermedia", *Proceedings of Hypertext '03 Conference, posters section*, 2003.

[204] D. Zhou, M. Truran, T. Brailsford, H. Ashman, and A. Pourabdollah, "Llama-B: Automatic Hyperlink Authoring in the Blogosphere", in *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. Pittsburgh, PA, USA: ACM, 2008.

APPENDIX A: A DATABASE DESIGN FOR ZZSTRUCTURE

A.1 Background

Zzstructure, which is a data structure being based on the new paradigm of ZigZag (more details on [131]) [110], uses different approach for structuring data that the conventional methods. The existing high power of relational databases is a motivating point for bridging between these two concepts. In this approach, zzstructure and relational database are used in two different level of abstraction and the solution consists of ideas in how to store and retrieve data in ZigZag paradigm by relations in fixed and simple structures. In this document, a provisional and an improved design for a client-server model based on such relational database are provided. The difference between two designs is an indexing method for providing easier and straight-forward access to the data in server side.

It is clear that the ZigZag paradigm is basically different from the way that relational database are normally used, but tables of a relational database can be used to store and retrieve data in zzstructure, because zzstructure and the proposed way of using database are in two different levels of abstraction [122]. While zzstructure is designed to be flexible and have no metadata [131], the main apparent obstacle for bridging between zzstructure and databases is structural strictness of table structures in the databases. From this point, one can assume that a database implementation of zzstructure leads to complexity and/or dynamic changes in tables' structures [122], [74]. This assumption is correct as long as one wishes to arrange zzstructured data directly to tables. Instead, this research uses databases with fixed structure as a query server which can produce a client's required views. Fixed but general and simple structure of a relational database is achievable when input and output requests are restricted to simple and fixed tables.

The fact that relational databases can be used for this purpose is also evidenced by noticing that:

Zzstructure can be defined as a directed multi-graph [117].

Zzstructure can be exported and imported to/from XML [129].

There are methods of building relational databases from both directed graphs and XML [33], [175], [77].

While current implementations of zzstructure don't include any relational database model [129], but an approach is used to bridge between XML, directed graphs and relational databases in [33] , [175], [77]. The first reference has used "ternary relation" which is "Generic relational schema for directed graph, independent of the XML schema" as:

Edge (source, label, destination)
Leaf (node, value)

The above relations are similar as the basic tables described in the next section. The difference is that label is explicitly typed into tables while it has been treated here by reference, same as source and destination. It is noticeable that the term "ternary relation" is more applicable to where three things of same type are related together than to the case of labelled graphs.

A.2. General Design

If tables of a relational database can be used both as input and as a required output, using the existing power of relational database can be helpful for building a zzstructure data server. Internal data arrangement in a relational database can enable it to store and retrieve data in zzstructure paradigm.

A complex zzstructure is dividable to the description of cells and links (while assuming dimensions as cells [129]), which both are describable with tables: Cells can be easily described by normal tables with multiple columns describing the properties of a cell. Each link in zzstructure also is a triple, consisting of an originating cell, a terminating cell and a dimension. Thus a general table design, with these three attributes is enough for describing the whole links. Seeing dimensions as cells, the records of Links table are triples of same objects. Obviously, a relation between the first and the last table can establish a referencing space with minimized redundancies. This architecture can obviously satisfy all normal forms.

Multiple visualizations of the same data are one of the most powerful points of zzstructure. Because of visual restrictions of being in 2 or 3 space dimensions, a visual system for zzstructure data must have a mechanism for assigning zzstructure dimensions to 2 or 3 space dimensions. Thinking in 2D space, two groups of visualizations are being use as H-view and I-view [129] (different terminology from views in relational databases). These two groups use a specific cell as the cursor, and then expand the view around it by the priority of horizontal or vertical dimension [117]. From this point, one of the common requirements of a zzstructure viewer system is retrieving ordered lists of cells in a given dimension (rank) which are around a specified cell, and this can be generated by a database server. This simplification, leads to this key point: Although visualization of a whole zzstructure data is possible by tables, but a specific views of a zzstructure can be expressed as views.

A.3 Detailed Design

This design consists of 2 tables of 'Cells' and 'Links'. Primary keys are 'cell_id' and 'link_id' respectively. Cells table is the place to store each cell as a record and has other descriptive fields (from a textual description to pointers to their multimedia contents). Links table is the place to store each ZigZag directional link as a record, which must contain the dimension (here, the dim_id, reference to the cell which stores that dimension), and the source and destination of the link as left_id and right_id (references to the cells of source and destination sides of a link, respectively). This is shown in Table A-1.

Table A-1: Basic tables' scheme

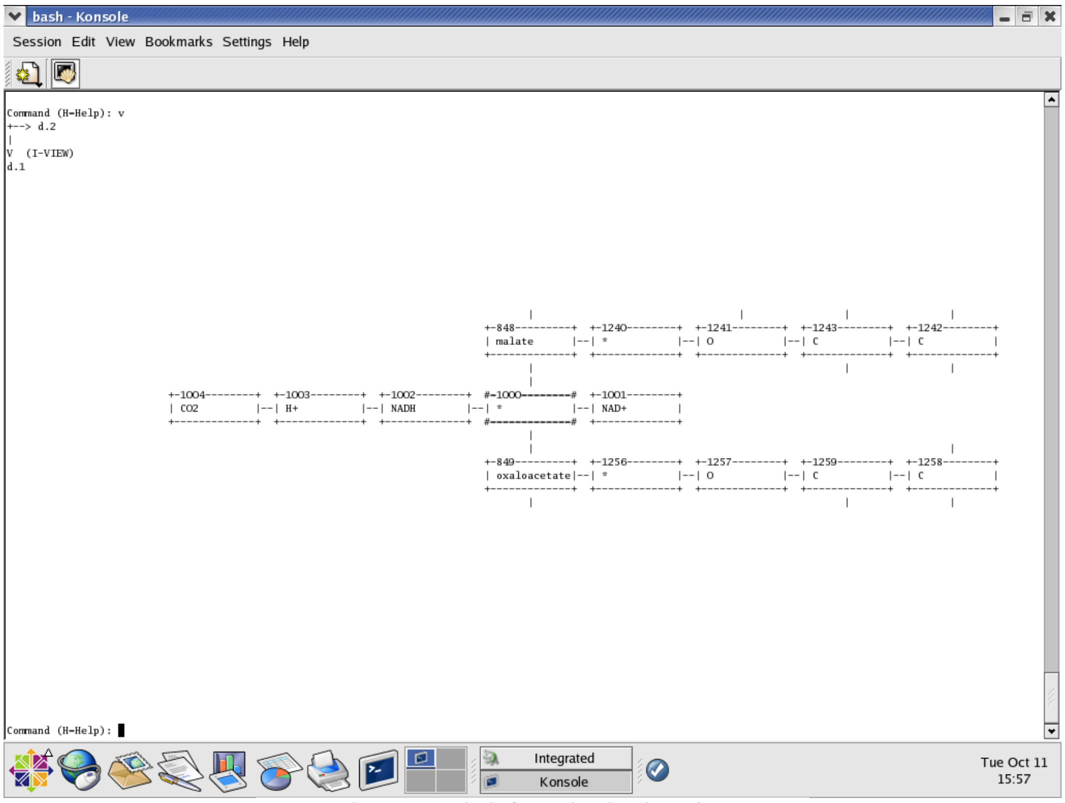
Cells	
cell_id	Other Descriptive fields

Links			
link_id	right_id	left_id	dim_id

A simple client action is looking for cells which are linked to a specified cell. The retrieval of the above structure is based on the finding out the left or right adjacent of a specified cell along a specified dimension. Looking forward, the client simply asks the server to retrieve the right_id of a link with specified left-side cell along a specified dimension in Links table, or vice versa for looking backward. After finding out an id, other specifications of a cell are retrievable using reference to Cells table.

A client-server prototype is designed which consists of using the above design in server-side using mySQL, and using C++ in client-side programming. The goal was testing the ability of such client-server architecture to exchange data in zzstructure. The minimum expected functionality was the ability of client to store, modify and navigate data, being defined as follows:

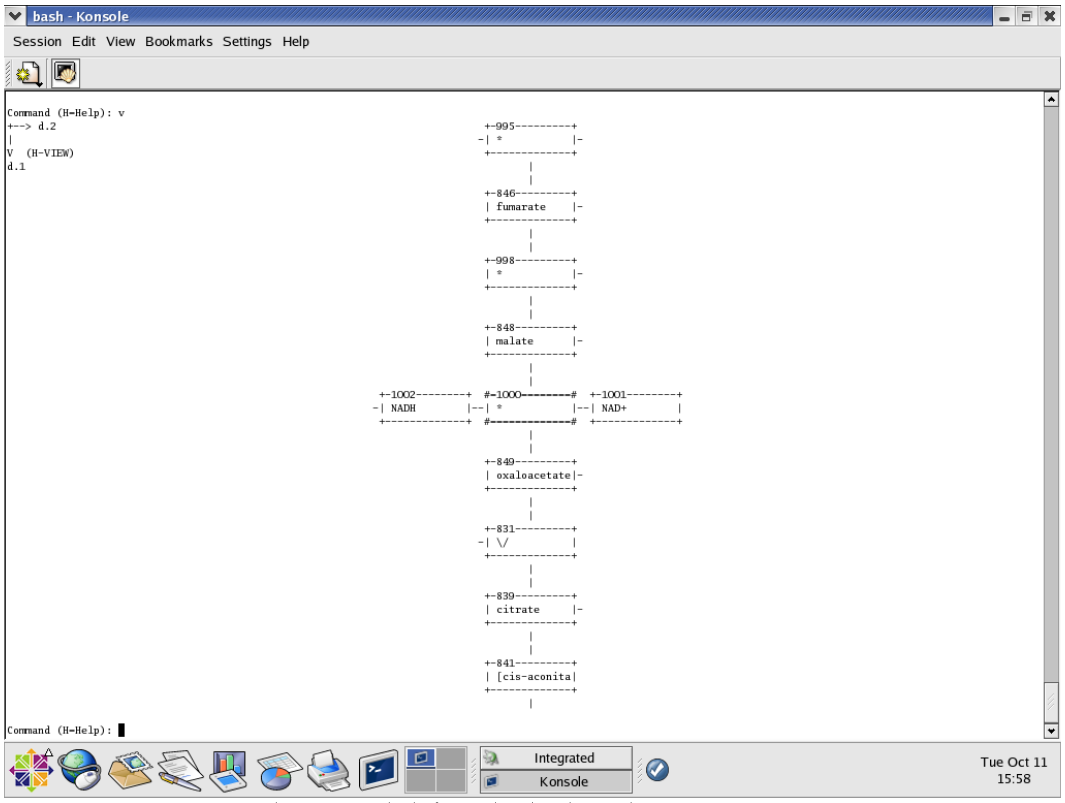
1. Storage: The ability of;
    1.1. Getting a text and define it as a cell (or dimension)
    1.2. Linking two predefined cells along a predefined dimension
2. Modification: The ability of;
    2.1. Changing the text of a cell
    2.2. Deleting a specified cell or a specified link
3. Navigation: The ability of;
    3.1. Defining the navigation dimensions as X and Y space dimensions.
    3.2. Moving a cursor pointer from a cell to one of its adjacencies along X or Y.
    3.3. Visualizing all cells and links around a cursor cell, in V- or H-view.
    3.4. Visualizing 2-D views both as V- or H-view.
    3.5. Searching for a specified text or id and make the found cell as the cursor.



```
Command (H-Help): v
+--> d.2
|
V  (I-VIEW)
d.1




                                                          |                                 |                |                |
                                                   +-848---------+  +-1240--------+  +-1241--------+  +-1243--------+  +-1242--------+
                                                   | malate      |--| *           |--| O           |--| C           |--| C           |
                                                   +-------------+  +-------------+  +-------------+  +-------------+  +-------------+
                                                          |                                                  |                |
                                                          |
+-1004--------+  +-1003--------+  +-1002--------+  #-1000--------#  +-1001--------+
| CO2         |--| H+          |--| NADH        |--| *           |--| NAD+        |
+-------------+  +-------------+  +-------------+  #-------------#  +-------------+
                                                          |
                                                          |                                                                   |
                                                   +-849---------+  +-1256--------+  +-1257--------+  +-1259--------+  +-1258--------+
                                                   | oxaloacetate|--| *           |--| O           |--| C           |--| C           |
                                                   +-------------+  +-------------+  +-------------+  +-------------+  +-------------+
                                                          |                                                  |                |




Command (H-Help): █
```

Figure A-1: Bio-informatics data in I-view

After implementing, the system has been tested successfully with real bio-informatics data described in [121, 122]. The data imported to the tables of the system using GZZ exported file (in XML format), which consists of about 2000 links. Figure A-1 and Figure A-2 show two screen-captures of the system when a part of the bio-informatics data is being visualized in H and I view.



```
Command (H-Help): v
+--> d.2                      +-995---------+
|                            -| *           |-
V   (H-VIEW)                  +-------------+
d.1                                  |
                                     |
                              +-846---------+
                              | fumarate    |-
                              +-------------+
                                     |
                                     |
                              +-998---------+
                              | *           |-
                              +-------------+
                                     |
                                     |
                              +-848---------+
                              | malate      |-
                              +-------------+
                                     |
                                     |
             +-1002--------+  #-1000--------#  +-1001--------+
            -| NADH        |--| *           |--| NAD+        |
             +-------------+  #-------------#  +-------------+
                                     |
                                     |
                              +-849---------+
                              | oxaloacetate|-
                              +-------------+
                                     |
                                     |
                              +-831---------+
                             -| \/          |
                              +-------------+
                                     |
                                     |
                              +-839---------+
                              | citrate     |-
                              +-------------+
                                     |
                                     |
                              +-841---------+
                              | [cis-aconita|
                              +-------------+
                                     |
Command (H-Help): 
```

Figure A-2: Bio-informatics data in H-view

The retrieval method in navigation process is based on of a single record in each step. While navigation, it uses SQL commands to retrieve a single-record view consisting an adjacent of a cell. For visualizing n cells in the screen, the system has to invoke at least n lookups, and for each lookup, a two-way communication with the server is necessary. As mentioned in the introduction, it is desired to use views as output, which is not fully realized with single record views. Thus it is more preferred to see a rank directly as consequent records of a view.

A.4. An Improved Design

The next step is to provide ways to directly retrieve ranks as views. Ranks are sequences of cells connected directionally along a certain dimension and views are filtered, joint, sorted tables that are extracted from the original tables. Sorted views have been used, because in the relational database theory records of a table are members of a set with no sequencing, while a rank has sequence inside. Because of this essential difference, a view can be a representation of a rank, only if it has an enforced sort by one of its fields.

Since it is preferred to retrieve a single rank as a view, filtration is necessary to distinguish between ranks to keep the cells in the same rank together. It is also necessary to find out membership of cells in the ranks. This is specially required when ranks are being used for grouping or cloning concepts and it is necessary to find out whether two cells belong to a rank or not.

Although answers for the above queries are implicitly embedded in the records of the basic tables, but none of them are explicitly visible by normal queries. For example, membership of two cells to a rank is not clear unless one can trace the links records for reaching from one cell to another.

Based on the above reasons, it is necessary to add two other fields to the Links table as index fields, one for distinguishing ranks and the other for sorting inside a rank. The associated cost is having more storage space and more processing load for re-indexing these fields in data entry stage. The solution is valid if an algorithm for re-indexing can be found. Two fields to be added to links table are: sort_id and rank_id, as in Table A-2.

Table A-2: The extended version of the Links table

Links					
link_id	left_id	right_id	dim_id	*sort_id*	*rank_id*

Because the server output is limited to be linear lists, a simple expression of loop ranks is a list of cells with repeated values in the start and the end. Although this repeated cell can be any cell in the loop, but client must tell the server its preferred starting cell (as the cursor). If the server's reply consists of starting cell at the end, the client will then recognize the rank as a loop rank; otherwise, it can be treated as linear rank.

If correct indexing in Links table can be done, then it is simple to create an intermediate view (say "pre-rank") by a single SQL statement as:

```
SELECT * FROM Links WHERE rank_id=… ORDER BY sort_id;
```

The above view must have a specific property that right_id in each record must be equal to left_id in the next record, excluding the last record. In addition, if it is a loop, right_id in the last record must be equal to left_id in the first record. Thus, the column of left_id's will be close to the desired table, but it doesn't include the last cell of the rank. This last cell appears in the column of right_id, instead. By using UNION query, the desired table can be built as another view by:

```
SELECT left_id AS cell_id FROM Pre-Rank UNION SELECT
LAST(right_id) AS cell_id FROM Pre-Rank;
```

Notice that the above view will work correctly for loop ranks, because it will result equal cell_id's for the first and the last records.

The solution has to be accompanied by an optimized re-indexing algorithm. Optimized means minimum data transfer between server and client. Also it is preferred to have no server side programming more than pure and standard SQL. This is not only because many database servers don't support internal programming and the solution shall be as general as possible, but also to avoid the probable time consumption required for running algorithm in programs. Re-indexing for these two fields means making sure that all links in a rank will share a unique rank_Id while each link in a rank has a unique sort_Id indicating its priority in that rank.

As usual, the indexing mechanisms are useful when there are more retrieving requests to the server that the modification requests. Indexing mechanisms cause better retrievals when sorting or searching is necessary in data retrievals. They shift and distribute the processing load to the data modification stage and free up the retrieval stage, by the cost of storage space for index keys. Although the relational databases use their own indexing mechanisms when a field needs to be indexed, but they may not help for this specific purpose.

A re-indexing algorithm is proposed which includes all possibilities of data modification and provides SQL statements for each situation. The main point in this algorithm is using inline (non-iterative) SQL statements. The details of this algorithm will be provided here.

A.5. Indexing Algorithms

This section includes details of an indexing algorithm to be used in conjunction with the method described in previous section. The goal of the indexing is to facilitate the access to the information stored in zzstructure. Since the described method in uses relational databases for data manipulation, the implementation of this algorithm has been intentionally limited to use plain SQL statements. This means that for all updating event, SQL statements will do all of the required tasks including re-indexing processes.

Details of the developed algorithm have been explained in this section after an example.

Here a simple example of a zzstructure arrangement is provided and how to store and index it using the described scheme will be shown. Let us suppose the cells are arranged as in Figure A-3. The equivalent link table is like Table A-3.

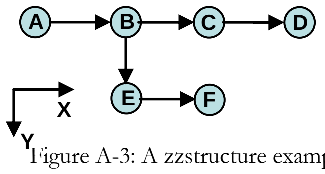


Figure A-3: A zzstructure example

Table A-3: Tabular representation

Links					
link_id	left_id	right_id	dim_id	*sort_id*	*rank_id*
1	A	B	X	1	1
2	C	D	X	3	1
3	B	E	Y	1	2
4	E	F	X	1	3
5	B	C	X	2	1

In Table A-3 the letters A, B, etc. are assumed to be Id's of the relevant cells. Also for generality, the links are added to the system not necessarily in their apparent sequences. Also in Table A-4 sort_id and rank_id are produced using the developed algorithm mentioned. Let's suppose that the client wants to retrieve the whole rank along dimension X that node B belongs to. First a simple query can extract that the required rank has rank_id=1. Then the view of 'Pre-rank' will filter it by rank_id=1 and sort it by sort_id and will result Table A-4. Finally a union query will result Table A-5, which is a ready-to-use table according to the mentioned principles.

Table A-4: View of Pre-Rank

Pre_Rank: SELECT * FROM Links WHERE rank_id=1 ORDER BY sort_id;					
link_id	left_id	right_id	dim_id	*sort_id*	*rank_id*
1	A	B	X	1	1
5	B	C	X	2	1
2	C	D	X	3	1

Table A-5: View of final Rank

Rank: SELECT left_id AS cell_id FROM Pre-Rank UNION SELECT LAST(right_id) AS cell_id FROM Pre-Rank;
cell_id
A
B
C
D

When a data entry or modification happens, necessary changes may be required on the index keys. In this section, the database is supposed to contain existing data with valid indices and the goal is to apply necessary changes on sort_id and rank_id after a single modification. Because of the reason behind of creation of these indices, the changes are successful as long as having two following conditions:

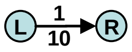
1. Having same rank_id for all links of a rank

2. Having no similar rank_id for links of any two different ranks

3. Increasing values of sort_id along consequent links of a rank

The goal of this algorithm is to provide an SQL-based indexing algorithm after such modifications. Being SQL-based here means that all of the necessary changes on the above tables must be done by consequent SQL statements that are being controlled by client application but being run on the server side. Since the server side is a general SQL server, it is preferred to do everything just by standard SQL. It is also noticeable that the provided SQL statements may not be ready to execute, because they may contain variable names that must be substituted with real ones. Further notice on this point will be described after describing the algorithm.

In the rest of this appendix, the notation of sort_id and rank_id will be two numbers over and under a link line, respectively. For example, if sort_id=1 and rank_id=10, then the link will be illustrated as:



All type of data entry or modifications in a zzstructure can be categorized in the following sections. For each section the algorithm and SQL statements for necessary changes on the index keys will be described.

Adding a New Link

A link wants to be added from cell with id=L to the cell with id=R along a dimension with id=D. For simplicity, here the existence of loops in ranks is ignored for the time being but will be considered later.

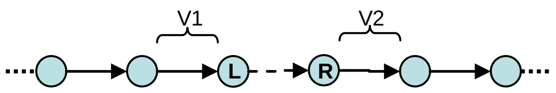
a) Required views:

View V1 indicates any link along D having L as its right side (left side business). In other word, it is the last link of any existing rank along D ending with L.

```
CREATE VIEW V1 SELECT * FROM Links WHERE
right_id=[L] AND dim_id=[D];
```

View V2 indicates any link along D having R as its left side (right side business). In other word, it is the first link of any existing rank along D starting with R.

```
CREATE VIEW V2 SELECT * FROM Links WHERE
left_id=[R] AND dim_id=[D];
```



The building link is noted as a dashed line.

View M1 indicates the maximum assigned value of rank_id in the Links table:

```
CREATE VIEW M1 SELECT MAX(rank_id) AS max_rank
FROM Links;
```

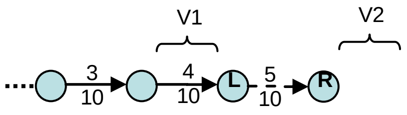
b) Finding out whether or not the left side and/or the right side of the adding link are busy

1) None of the sides are busy; i.e. V1 and V2 are empty. No change on the previous records is required, just adding the new link by sort_id=1 and a new value for rank_id. Because it is enough for the value of rank_id to be unique, this new value can simply be one unit more than the maximum value of rank_id's in Links table. In the future, when this rank expands, this rank_id for the other added links will be copied.

```
INSERT INTO Links (left_id, right_id, dim_id,
sort_id, rank_id) SELECT [L] AS c1, [R] AS c2,
[D] AS c3, 1 AS c4, max_rank+1 AS c5 FROM M1;
```

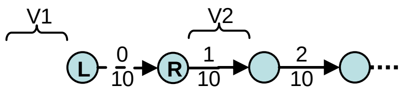
2) Only the left side is busy; i.e. V1 is not empty while V2 is empty. In this case, the building link is at the end of a rank, so the rank_id can be gotten from the left side link and the sort_id is one unit more than the last one:

```
INSERT INTO Links
(left_id,right_id,dim_id,sort_id,rank_id) SELECT
[L] AS c1, [R] AS c2, [D] as c3, V1.sort_id+1 AS
c4, V1.rank_id AS c5 FROM V1;
```



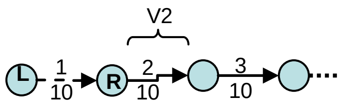
3) The right side is busy; i.e. V1 is empty while V2 is not empty. In this case, the building link is at the beginning of a rank, so the rank_id can be gotten from the right side link. About sort_id, it can simply be one unit less than sort_id of V2 (when negative numbers are allowed).

```
INSERT INTO Links (left_id, right_id, dim_id,
sort_id, rank_id) SELECT [L] AS c1, [R] AS c2,
[D] AS c3, V2.sort_id-1 AS c4, V2.rank_id AS c5
FROM V2;
```



To avoid negative numbers, all of sort_id's in the left side rank must be incremented constantly. Notice that now the minimum possible sort_id is zero, which is normally allowed for unsigned integers, but the problem can occur for the next adding links. Thus, a simple way is to increment sort_id's in all links of this rank by:

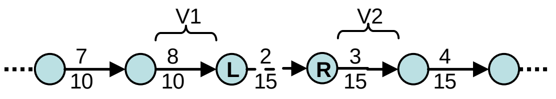
```
UPDATE Links, V2 SET
Links.sort_id=Links.sort_id+1 WHERE
Links.rank_id=V2.rank_id;
```



The last SQL is a serial updating which can slow down the speed of the algorithm for a common case, but it is still an option to keep sort_id as unsigned integer and doing the above procedure. In the rest of this document, signed integer will be used for sort_id, thus the last SQL statement is ignored.

4) Both sides are busy; i.e. both V1 and V2 are not empty. In this case, two separate ranks are going to join. First, a same procedure as the previous paragraph is necessary to insert the new link:

```
INSERT INTO Links (left_id, right_id, dim_id,
sort_id, rank_id) SELECT [L] AS c1, [R] AS c2,
[D] AS c3, V2.sort_id-1 AS c4, V2.rank_id AS c5
FROM V2;
```

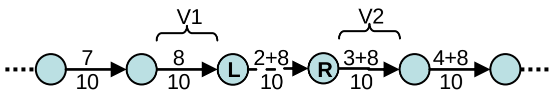


Because the existence of an invalid breakdown is probable on the cell L, it is necessary to update sort_id's in one side of this cell. Although updating on the side with less number of ranks is beneficial, but counting the number of ranks can itself be time consuming. Studying this case is up to practical measurements on real data, but for the time being, right side is selected arbitrarily. In this case, updating the right side sort_id's is adding the maximum sort_id in the left side to all of them. This maximum is on the link V1, because V1 is the last link of the left side. The criteria of belonging to the right side is clear by filtering on rank_id to be equal to [V1.rank_id] which automatically includes the new link, because it is already added by this rank_id.

On the other hand, L is also a discontinuity point for rank_id and rank_id's must be updated in one side. The same considerations (like the last paragraph for selecting which side) are applicable here, but the final decision must be the single side for both, because all the changes are kept on a single side which has a higher priority. Thus the right side is selected and all rank_id's must be changed to [V1.rank_id].

Thus, a single SQL statement can re-index sort_id's as well as rank_id's:

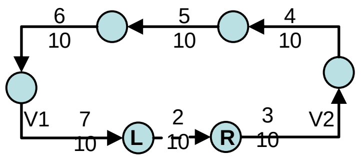
```
UPDATE Links,V1 SET
Links.sort_id=Links.sort_id+V1.sort_id,
Links.rank_id=V1.rank_id WHERE
Links.rank_id=V2.rank_id;
```



c) Considering the case of loops in ranks

A loop can be built only in section 4 above (both sides busy). Things to be reconsidered are:

1) When adding a new link, business in both sides will no longer means joining different ranks, because the new link can be the last link of a loop. However, the SQL statements in that section are still valid. Notice that [V1.rank_id]=[V2.rank_id] (because V1 and V2 belong to a single rank) and the SQL statement will overwrite all rank_id's with their previous values.



2) If a loop is constructed, sort_id's along the rank after execution of SQL statements will be increasing numbers with a breakdown before or after the added link. This will not cause any problem because sorting the links when the server replies the queries will produce a list starting or ending with the added link.

Consequently, this will produce a cell list starting and ending with a same cell, which is the meaning of a loop.

d) Unifying all cases

As shown before, considering the case of loops will keep the validity of the algorithm. However, the algorithm can do some redundant sweeping along the ranks without any effect. Since a general algorithm is preferred to be run for adding a link, the following algorithm can be executed:

```
If V2 is empty
   If V1 is empty
      INSERT INTO Links (left_id, right_id,
      dim_id, sort_id, rank_id) SELECT [L] AS c1,
      [R] AS c2, [D] AS c3, 1 AS c4,
      MAX(rank_id)+1 AS c5;
   Else
      INSERT INTO Links
      (left_id,right_id,dim_id,sort_id,rank_id)
      SELECT [L] AS c1, [R] AS c2, [D] as c3,
      V1.sort_id+1 AS c4, V1.rank_id AS c5 FROM
      V1;
   End If
Else
    INSERT INTO Links (left_id, right_id, dim_id,
    sort_id, rank_id) SELECT [L] AS c1, [R] AS
    c2, [D] AS c3, V2.sort_id-1 AS c4, V2.rank_id
    AS c5 FROM V2;

    If V1 is not empty
      UPDATE Links,V1 SET
      Links.sort_id=Links.sort_id+V1.sort_id,
      Links.rank_id=V1.rank_id WHERE
      Links.rank_id=V2.rank_id;
    End If
End If
```

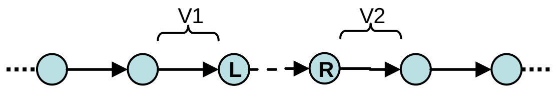
Breaking a link

A link between cell with id=L to the cell with id=R along a dimension with id=D needs to be broken. Here also for simplicity, the case of loops is temporarily ignored and will be studied later.

a) Direct deletion

First, the specified link can directly be deleted by the following command:

```
DELETE * FROM Links Where left_id=L AND
right_id=R AND dim_id=D;
```



The deleting link is noted as a dashed line.

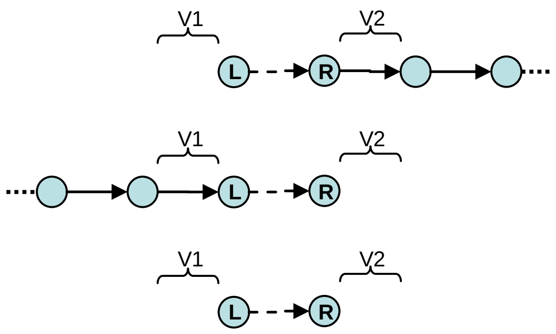
The next steps are updating index keys in other links of the rank.

b) The required views

They are V1 and V2 and M1 as in the algorithm for adding links. Two other views V3 and V4 are also necessary but will be defined later.

c) Finding out whether or not the left side and/or the right side of the breaking link are busy:

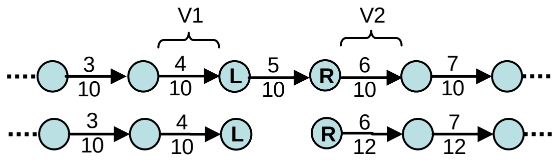
1) Only one side is busy or none of sides are busy; i.e. V1 or V2 is empty. In these cases, there is nothing to do because either the first or the last link of a rank or a single-link rank has been deleted. In the first case, the rest of the rank has already been re-indexed before, so no changes are required on the previous links. In the second case, a whole rank has been deleted and there is no other link to be re-indexed.



Both sides are busy; i.e. V1 and V2 are not empty. In this case, a single rank has been split to two separate ones. Although sort_id's of the two side ranks can remain unchanged (because they have already correct sorting index) but it is necessary to change rank_id's in at least one of the produced ranks to be differentiated from the other. This new value has to be a new one without any previous usage. A simple method is to change rank_id in one side by a new value.

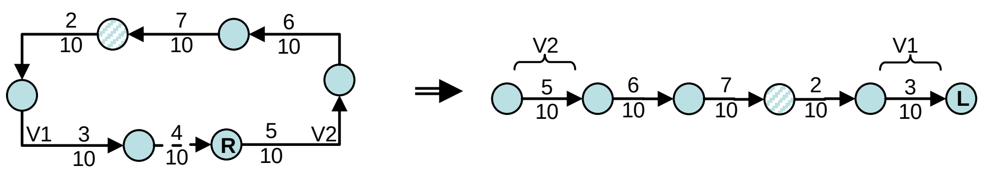
Although doing the procedure on the side with less links is beneficial, but counting the number of links in each side can be time consuming. This trade off can be studied in practice, but in this document, the right side will be modified for simplicity. The new value of rank_id in the right side can be equal to one unit more than maximum value of rank_id's in the Links table. To select the right side, one can use comparing sort_id's with V2.sort_id:

```
UPDATE Links,V2,M1 SET
Links.rank_id=M1.max_rank+1 WHERE
Links.rank_id=V2.rank_id AND
Links.sort_id>=V2.sort_id;
```



d) Considering the case of loop ranks:

Case 1 above cannot happen for loops, but in case 2, if link of a loop has been deleted, it has converted a loop rank to a linear rank, not two ranks. As described before, sort_id's in a loop cannot be an increasing value and must have a breakdown step in one of its cells. Deleting one of the links means probable leaving of this breakdown cell in the new linear rank, which will cause invalid indexing. The breakdown will not exist if one of the two links around that cell is occasionally deleted. In the later case, there will be nothing to do with sort_id's, but in general case, sort_id's must be re-indexed.



(The breakdown cell is highlighted as diagonal pattern.)

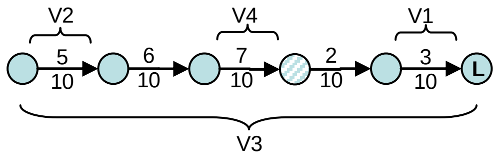
It will be shown how to re-index sort_id's in the next paragraphs, but after doing that, one may concern about rank_id's. The re-indexing algorithm for rank_id's described in case 2 must not make different rank_id's for links of the rank after breaking the loop. Fortunately, the SQL statements in that section are still valid. This is because once sort_id's are re-indexed, V1 will have the maximum value of sort_id and V2 will have the minimum value; i.e. for all rank, [sort_id]<=[V1.sort_id] and [sort_id]>=[V2.sort_id]. This means that SQL statement will sweep all rank, and the whole rank will have same rank_id's.

For re-indexing sort_id's, a sort_id breakdown point in all links with same rank_id as rank_id of V1 or V2 must be traced. If a loop around its breakdown point is going to be broken or if a linear rank is breaking into two, a breakdown point will not be found and the rest of algorithm has no effect. For this purpose, view V3 is created as:

```
CREATE VIEW V3 SELECT * FROM Links WHERE
rank_id=V1.rank_id;
```

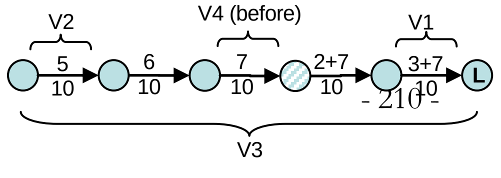
V3 for case of loop breaking means the produced linear rank, and for breaking the linear ranks means the whole left side rank. Notice that a breakdown point of sort_id in V3 can be defined as a cell which has sort_id on its left-side link greater than sort_id on its right side link. View V4 will contain a link in V3 which has such a breakdown on its right side cell. It will contain any breakdown existence in V3 by relating V3 to itself as:

```
CREATE VIEW V4 SELECT * FROM V3 AS T1 INNER JOIN
V3 AS T2 ON T1.right_id=T2.left_id WHERE
T1.sort_id>T2.sort_id;
```



If V4 is not empty, V4.sort_id is the maximum of sort_id's values in V3, because sort_id's in V3 are increasing values up to the breakdown point. The remaining step is to add this maximum value to sort_id's in the appropriate records of V3:

```
UPDATE V2,V3,V4 SET
V3.sort_id=V3.sort_id+V4.sort_id WHERE
V3.sort_id<V2.sort_id;
```



The criteria of sort_id<=V1.sort_id comes from the point that V2 is the link at the leftmost of the rank and any link with sort_id less that V2.sort_id needs to be added up to be sorted before V2.

Finally, if V4 is empty, which means no breakdown, there is no need to update any sort_id.

e) Unifying all cases:

Considering the case of loops will result execution of the last SQL statement before the SQL statements of case 2, as well as creating two views V3 and V4 defined above. As described, all of these consequent statements will work for both cases of loop ranks and linear ranks. The whole algorithm for deleting a link will be as follows:

```
If V1 is not empty AND V2 is not empty
    UPDATE Links,V2,M1 SET
    Links.rank_id=M1.max_rank+1 WHERE
    Links.rank_id=V2.rank_id AND
    Links.sort_id>=V2.sort_id;

    If V4 is not empty
        UPDATE V2,V3,V4 SET
        V3.sort_id=V3.sort_id+V4.sort_id WHERE
        V3.sort_id<V2.sort_id;
    End If
End If
```

Inserting a Link

This is possible by consequent actions of breaking a links and adding two others.

Adding or Modifying a Cell

No changes on indexes required.

Deleting a Cell

Deleting a cell is not allowed while it is involved in a link. This restriction can be carried out by enforcing the rules of referential integrity, which is defined when creating the main tables of cells and links. If all the involved links are deleted with the described algorithm, deleting a cell will not affect any index key in links table.

1) The mentioned unified algorithms for adding and deleting a link include variables L, R and D that must be substituted with their real value or correct references to them. For this purpose two alternatives are possible.

First, by direct substitution in the client side when invoking. The application in the client side can build and send the substituted statements. The problem with this solution is that for each modification, it is necessary to build and overwrite the required views (like V1,V2, etc.) repeatedly, which can be time consuming.

Second, by feeding from a table: It is possible to add a single record table for temporary storing the values of L, R, and D. The client side application must store the real values in that table with simple SQL statements. In this case, one has to apply necessary changes to SQL statements to select and use the values from that table. For example, let's have a new table called Config with this structure:

Config			
L	R	D	Other fields

Supposing that this table has a single record, the client will "build" and send an SQL statement like: UPDATE Config SET L=1, R=2, D=3; Then simple changes like the following is necessary for any SQL statement which uses L, R, or D:

The example before change:

```
INSERT INTO Links (left_id, right_id, dim_id,
sort_id, rank_id) SELECT [L] AS c1, [R] AS c2,
[D] AS c3, 1 AS c4, MAX(rank_id)+1 AS c5;
```

And after change:

```
INSERT INTO Links (left_id, right_id, dim_id,
sort_id, rank_id) SELECT Config.L, Config.R,
Config.D, 1 AS c4, MAX(rank_id)+1 AS c5 FROM
Config;
```

If the second method is used, it is also possible to build the necessary views (V1,V2, etc.) once and in advance. For example, view V1 can be built in the database independent of the client application by:

```
CREATE VIEW V1 SELECT * FROM Links, * FROM Config
WHERE Links.right_id=Config.L AND
Links.dim_id=Config.D;
```

2) The algorithms for adding or deleting a link contain If-blocks. For implementing these if-blocks, there are at least three alternative solutions: First, to use server side programming: This is not recommended because it was intended to use standard and simple SQL server, which may not necessarily support server side programming. Second (not recommended), to use internal function of SQL (like IIF() function): Some of the SQL statements above can be unified using IIF() may increase their complexity. To remove all of the if-blocks, it will add much complexity. Third, to use programming (or scripting) language in the client side: This is more recommended. Although a decision made in the client side, but does not violate the initial intension of doing processes in server side. This is because the decision will not imply any unnecessary data exchange between client and server. The only investigated thing in client side is checking a view to be empty or not.

APPENDIX B: NOTES ON ONE-TO-MANY RELATIONSHIP IN ZZSTRUCTURE

B.1. Background

It has been mentioned in section 2.3.1 that there is no direct solution exists for one-to-many relationship representation, because of fundamental rules of ZigZag. This document studied the associated problems of indirect solutions of representing one-to-many relationship by zzstructure. A main source of these problems is that grouping in ZigZag cannot be done without sequencing. In addition, these problems will be considered in the designed relational database for storing zzstructured data (described in Appendix A). Finally, three approaches will be provided to address the associated problems. The last provided approach is adding an extra concept to the ZigZag paradigm (which is Macro-cell), and the result will no longer be zzstructure.

It is necessary to provide a working definition for one-to-many relationship: *Linking a single cell, equally to other cells, by a single relation, in a single context.* It is noticeable that the many-side is a set of cells, without any sequence. This definition can cover two interesting applications in this study: Data grouping and Cloning. Data grouping is where it is needed to put some cells in a set without sequencing. In this case, data grouping is a relation between a set's name and its members. Cloning is one of the mechanism in zzstructure to simulate one-to-many relationship, in which several instances of cell are used to relate to other cells, while themselves are linked through a special dimension: d.clone [129]. It is a one-to-many relationship because it is a relation between a cell and its clones, noticing that clones of a cell have no point in sequencing. However, if any sequencing is required between clones, it can be done via another dimension

B.2. One-to-Many relationship Mechanisms

Because of basic limitations of ZigZag topology to represent one-to-many relationships, indirect topology must be used for this purpose. Some of such solutions are represented in Figure B-0-1.

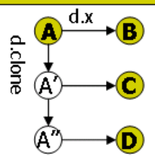
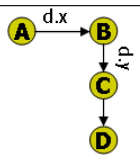
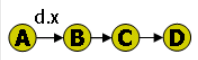
	Topology	External Convention	Redundant/ Unreal Info.
Cloning		d.clone meaning	- Extra Cells - Sequencing - Different Context
Pan		d.y meaning	- Sequencing - Exclusive relation
Head Cell		Head cell meaning	- Sequencing - Exclusive relation

Figure B-0-1: Indirect Solutions of Zzstructure for One-to-Many Relationship

B.3. The Associated Problems

A problems source is that the topology may need external convention for being understood. Different meanings of structure can be understood with or without a convention (Uncertainty). Another source of problems is possible added information for satisfying ZigZag rules (Redundancy). The existence of redundancy and uncertainty can deviate the structure from the real information, as illustrated in Figure B-0-2.

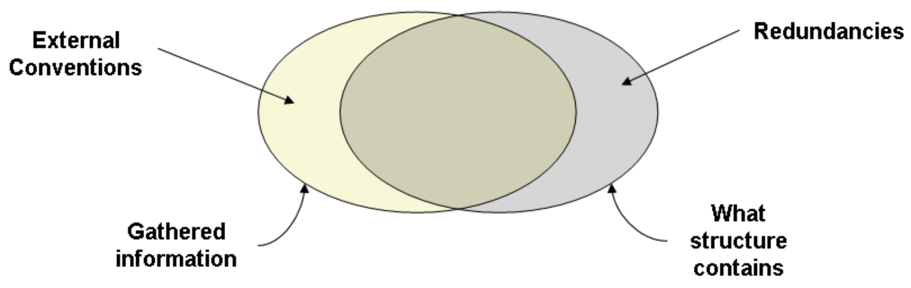


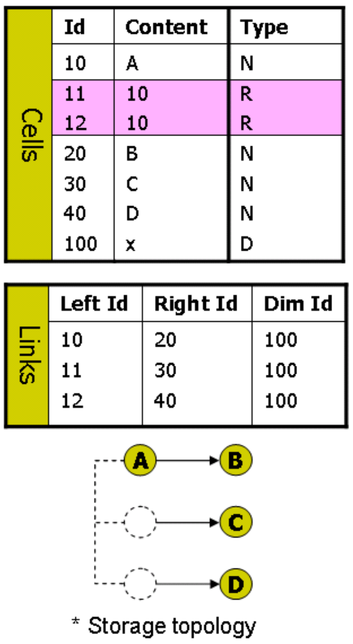
Figure B-0-2: Deviation of structure from information

Considering the provided design for client-server approach to storing zzstructure in relational databases (described in Appendix A), possible effects can appear in storing redundancies (because of resource seizure), in sequential access (low speed in search, sort,…) and in modification (multiple steps in data modification, which means lower speed and more risk of data loss).

The following approaches are possible solutions when some or whole design goes beyond ZigZag rules.

B.4. Storage-layer Optimization by Referencing Cells

This approach is similar to cloning mechanism for one-to-many relationship, but with no sequencing along d.clone. Figure B-0-3 illustrates the storage design of cloning with an example.



Cells

Id	Content	Type
10	A	N
11	10	R
12	10	R
20	B	N
30	C	N
40	D	N
100	x	D

Links

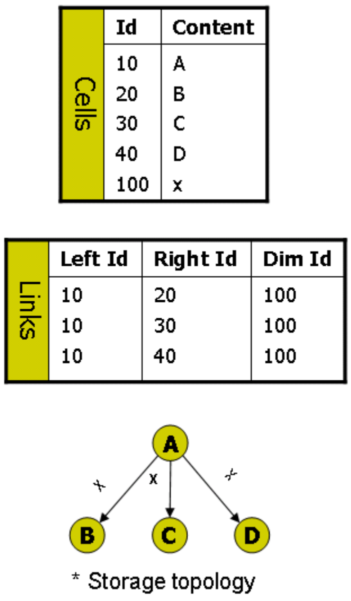
Left Id	Right Id	Dim Id
10	20	100
11	30	100
12	40	100

Figure B-0-3: Referencing-Cells Design

This solution also must be accompanied by adding a layer on the top of storage layer which contains suitable views to show us the data as expected by the client application. Effectiveness in search and data modification must be studied practically but it is clear that the required space is less than before and there is no sequence between clones which can provide random access to clone cells.

B.5. Storage-layer Optimization by Non-ZigZag Storage

This solution also must be accompanied by adding a layer of views to show us the data as expected by the client application. Illustration of an example in Figure B-0-4 clarifies this approach.



Cells	Id	Content
	10	A
	20	B
	30	C
	40	D
	100	x

Links	Left Id	Right Id	Dim Id
	10	20	100
	10	30	100
	10	40	100

Figure B-0-4: Non-ZigZag Storage

The storage explicitly breaks ZigZag rules but can provide better search and data modification on the row data on the cost of extra processing over storage layer. The issue of conversion process from/to zzstructure in query layer must be noticed, because both cells and links tables must be changed while the conversion must be optimized to be fast. The overall performance of such a design can not be simply estimated without practical implementations.

B.6. Changes on Fundamentals: Macro-Cells

This approach tries to add a fundamental concept to ZigZag: *Grouping*. A modified zzstructure is proposed, which is no longer ZigZag, but uses its rules and adds the concept of "Macro Cells" to it. In this structure, *cells can be member of each other*. A cell can be member of 0 or 1 another cell; i.e. a cell can not be member of two groups in a same context. Also all cells still are ZigZag cells, and can connect under ZigZag rules, regardless of their container or contents. This has been illustrated in Figure B-0-5.

The resulted structure is backward compatible to zzstructure. This means that it can still be used for storing zzstructured data. Cloning mechanism can have its previous implementation (i.e. connecting along d.clone and storing unnecessary sequencing), but it shall not be used for one-to-many relationship (because there will be a systematic way for doing that). As mentioned before, cloning is a one-to-many relationship between a main cell and its clones. Thus the clones are empty members of a main cell in this new structure (with no d.clone, no sequencing). Visualization of this new structure will be different from ZigZag. The suggestion is that a macro cell will be visualized in its context, regardless of context of its members. The vision can be transferred to its members' context by user's wish.

The storage of data in this new structure is possible by adding a "member-of" field to the Cells table, with no change to Link table. This has been shown in Figure B-0-6.

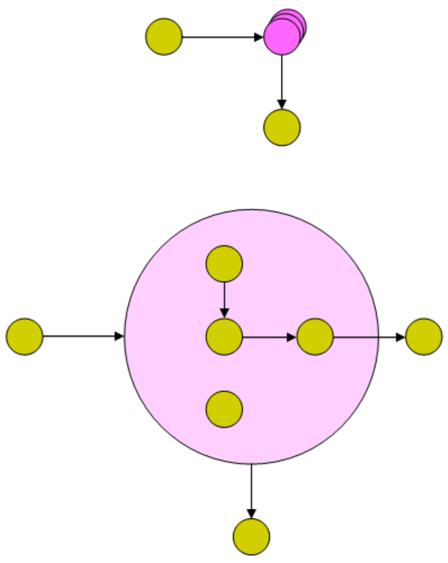
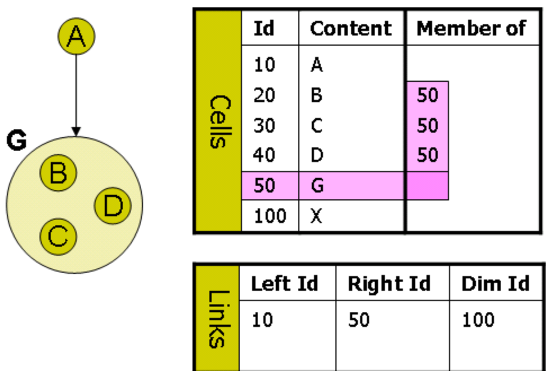


Figure B-0-5: Notation and Visualization of the Concept of Macro-cell



Cells	Id	Content	Member of
	10	A	
	20	B	50
	30	C	50
	40	D	50
	50	G	
	100	X	

Links	Left Id	Right Id	Dim Id
	10	50	100

Figure B-0-6: Data storage of Macro-cells

B.7. Section Summary

Because of indirection in the possible ZigZag representations for one-to-many relationships, a combination of uncertainty and redundancy can cause several problems in data manipulation. In the proposed client-server architecture for using relational database for zzstructure, these problems have been studied and these approaches have been described: Different storage methods in server side, and a new non-ZigZag structure which adds the concept of non-sequential grouping to Macro Cells.

Figure B-0-7 compares the number of required records in two table of storage layer, for different approaches. If n is the number of cells in many-side, it has been shown that the total number of records varies from a function of 4n in cloning mechanism to a function of n in Macro-cell structure. This can be used for justification of Macro-cell structure.

	Method	Storage Records			
		Cells	Dims	Links	Total
Pure-zz	Cloning	2n	2	2n-1	4n+1
	Pan	n+1	2	n	2n+3
	Head Cell	n+1	1	n	2n+2
Improved	Non-zz Storage	n+1	1	n	2n+2
	Referencing Cells	2n	1	n	3n+1
	Macro Cells	n+2	1	1	n+4

n: number of cells in *Many* side

Figure B-0-7: A Comparison between the storage seisure for

APPENDIX C: SOURCE CODES

The programming source codes related to the different parts of this thesis have been written to the attached CD.

The folders on the CD and the related sections are as follows:

1. Folder "TRM-DB" related to the listings of chapter 4.

2. Folder "TRM-NAV" related to section 5.5.1

3. Folder "TWM" related to section 7.1

4. Folder "ZZCLIENT" related to Appendix A

Also the file "TRM.sql" in the root directory is an exported file from the used mySQL database. It includes the necessary sample databases from other folders and shall be imported into the working mySQL database if necessary.

The list of the files and folders together with a short description of each one is listed in the following table.

A copy of the CD contents is also accessible online at http://cs.nott.ac.uk/~axp/thesiscd.

Root	
packinglist.doc	Includes this table
TRM.sql	Includes the sample data necessary to be imported to mySQL database on destination server
\TWM\public_html\workflow	
graph.html	Shows the workflow sample graph
index.html	Homepage of TWM
twm.jpg	The TWM logo
workflow.jpg	Includes the picture of workflow sample graph
\TWM\cgi-bin	
history.php	Produces the history page, i.e. the list of tasks applied on a certain workflow case. Parameter: c indicating the case ID
workflow.php	Produces the multi-frame home page of TWM, a page including topframe.php at top and a placeholder for other pages at the bottom.
startflow.php	Produces a page to guide the user to start a workflow case. Parameter: u for User ID
topframe.php	Produces a menu bar for the system to sit on top of the homepage
started.php	Starts a workflow case from an indicated workflow node and produces a page showing the necessary message to the user. Parameter: n for the starting node; c for the starting case
inbox.php	Produces the inbox page for each workflow users, showing a list of cases which requires the user’s action, together with the available actions to choose. Parameter: u for User Id
doaction.php	Performs the selected action in inbox and refresh the inbox page. Parameter: c: Case Id n: Node Id which contains the relation s: Source node Id a: Association node Id d: Destination node Id r: reverse action if r=1, ordinary otherwise.
ZZCLIENT	
main.cpp	The main C++ program to run the ZigZag client
\TRMNAV\js	
dw_viewport.js	The JavaScript code necessary to run within HTML files, mainly to provide floating sub-menus.
\TRMNAV\public_html	
trmlink.css	The CSS codes for the visual effects of the floating menu
index.html	The homepage showing the links to the original and modified pages
cssmenu.htm	Produces the floating menus
atom.html	A sample HTML file about atoms
electron.html	A sample HTML file about electrons
proton.html	A sample HTML file about protons
photon.html	A sample HTML file about photons
neutron.html	A sample HTML file about neutrons
electricity.htm	A sample HTML file about electricity
stimulation.htm	A sample HTML file about stimulation
excitation.html	A sample HTML file about excitation
trmlogo.jpg	The logo of TRM
\TRMNAV\cgi-bin	
readfile.php	Produces a new HTML having Ternary Links, by reading any other normal HTML and looking for the link anchors by accessing to the mySQL database. Parameter: u indicating the URL of the input HTML.

\TRMDB	
atomTRM.xml	The TRM-XML equivalent of TRM-NAV atom example
TRMBiblio.mdb	The mdb equivalent of the sample bibliography database, implemented in TRM-Table. In also includes sample SQL statements for querying TRM-Table.
sampleTRM.xml	A sample TRM-XML listing
sampleworkflow.xml	A sample TRM-XML equivalent for a workflow definition
TRM.xsd	The TRM-XML Schema
TRMBiblio.xml	A sample TRM-XML listing for the bibliography example
TRMBiblio.xql	A sample TRM-Query implemented in XQuery
TRM-noannotation.xsd	The TRM-XML Schema without annotations

APPENDIX D: THE PUBLISHED WORKS