

Knowledge Extraction Framework from Scripts, Use-cases, Knowledge Maps, Decision Tables and Decision Trees for Knowledge Warehouse

To 7635



Submitted by

Bilal Ahmed
257-FAS\MSCS\F05

Supervised by

Prof. Dr. Maqbool-uddin Shaikh

Department of Computer Science
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad

(2010)



Accession No. YH 7635

MS
005.74
BIK

Handwritten signature

1- Data mining



IN THE NAME OF
ALMIGHTY ALLAH
THE MOST BENEFICENT
THE MOST MERCIFUL

**Department of Computer Science
International Islamic University Islamabad**

08-Nov, 2010

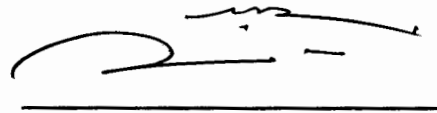
FINAL APPROVAL

It is certified that we have read the thesis submitted by Mr. Bilal Ahmed Reg. No. 257-FAS/MSCS/F05 and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the MS Degree in Computer Science.

COMMITTEE

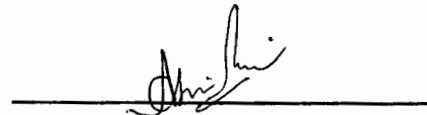
External Examiner

Prof. Dr. Jamil Ahmad
IQRA University,
Islamabad.



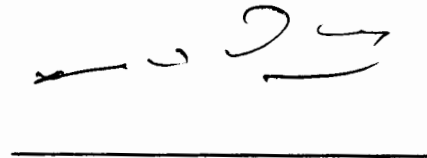
Internal Examiner

Mr. Asim Munir
Department of Computer Science,
International Islamic University,
Islamabad.



Supervisor

Prof. Dr. Maqbool-uddin Shaikh
Computer Science Department
COMSATS Institute of Information Technology
Islamabad.



**A Dissertation Submitted to the
Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University, Islamabad, Pakistan,
as a partial fulfillment of the requirements for the award of the degree of
MS in Computer Science**

***DEDICATED
TO
MY FAMILY***



DECLARATION

I hereby declare that this thesis, neither as a whole nor as a part thereof has been copied from any source. It is further declared that I have developed this thesis entirely on the basis of my personal efforts made under the sincere guidance of my supervisors. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Bilal Ahmed
257-FAS/MSCS/F05

Acknowledgment

First and foremost, I'd like to thank Almighty Allah for choosing me for this work.

Secondly I would like to thank my pappa who is always encouraging and supporting me to complete my thesis.

Then I would like to thank my supervisor Prof. Dr. Maqbool-ud-Din Shaikh.

My special Thanks to my teachers Mr. Imran Saeed and Mr. Asim Munir. They have respect of as my best teachers but frank as my closest friends.

Abstract

This thesis proposes a framework for knowledge extraction and transformation for a knowledge warehouse on the same line as the concept of ETL in data warehouse. A storage structure for knowledge warehouse is also proposed. The ETL framework will be used to device algorithms for knowledge extraction and transformation from five types of knowledge documents scripts, use cases, knowledge maps, decision tables and decision trees. These algorithms extract knowledge from these five types of knowledge documents and store them in the proposed storage structure of a knowledge warehouse. A case study of each algorithm to visualize the working is also conducted and included in this thesis.

*Keywords: Knowledge warehouse, Knowledge ETL, Knowledge Extractions
Algorithm, Warehouse Framework*

Table of Contents

Chapter 1	Introduction	2
	1.1 Motivation and Challenges	3
	1.2 Background	4
	1.3 Research / Problem Domain	5
	1.4 Proposed Approach	6
	1.5 Thesis Outline	6
Chapter 2	Literature Survey	7
	2.1 Introduction	8
	2.2 Related Research / Technologies	16
	2.3 Limitations	19
	2.4 Summary of Chapter 2	19
Chapter 3	Requirement Analysis and System Design	21
	3.1 Introduction	22
	3.2 Problem Scenario	22
	3.3 Focus of Research	23
	3.4 Design Requirements	23
	3.5 Reference Architecture	23
	3.6 Proposed ETL framework for knowledge warehouse	25
	3.7 Proposed Node structure for knowledge warehouse	29
	3.8 Summary of Chapter 3	32
Chapter 4	Knowledge Extraction Algorithm for Decision Tree	33
	4.1 Introduction	34
	4.2 Literature Review	34
	4.3 Problem Scenario for Decision tree ETL algorithm	36
	4.4 Proposed Algorithm for ETL from Decision Trees	37
	4.5 Case Study for proposed ETL Algorithm	38
Chapter 5	Knowledge Extraction Algorithm for Decision Tables	41
	5.1 Introduction	42
	5.2 Literature Review	42
	5.3 Problem Scenario for Decision Table ETL Algorithm	51
	5.4 Proposed Algorithm for ETL from Decision Tables	51

	5.5	Case Study for proposed Algorithm	54
Chapter 6		Knowledge Extraction Algorithm for Use Cases	57
	6.1	Introduction	58
	6.2	Overview of use case documents	58
	6.3	Literature Review for use case algorithms	61
	6.4	Problem Scenario for use case algorithms	63
	6.5	Proposed Algorithm for ETL from Usecases	64
	6.6	Case Study for proposed algorithms	65
	6.7	Implementation of algorithm	68
Chapter 7		Knowledge Extraction Algorithm for Knowledge Maps	69
	7.1	Introduction	70
	7.2	Literature Review	70
	7.3	Problem scenario for knowledge ETL Algorithm	75
	7.4	Proposed Algorithm for ETL from knowledge maps	77
	7.5	Case Study for proposed algorithm	78
	7.6	Implementation of algorithm	81
Chapter 8		Knowledge Extraction Algorithms for Scripts	82
	8.1	Introduction	83
	8.2	Literature Review	83
	8.3	Proposed Algorithm for Knowledge ETL from Scripts	88
	8.4	Case Study of proposed algorithm	89
Chapter 9		Conclusion and Outlook	91
	9.1	Introduction to the outcome of the thesis	92
	9.2	Achievements	92
	9.3	Improvements / Enhancements	94
	9.4	Future Work and Recommendations	94
	9.5	Summary of the thesis	95

List of Figures

Figure 2.1	The activities of Business Intelligence	9
Figure 2.2	Knowledge Spiral	11
Figure 2.3	Knowledge Warehouse Architecture	13
Figure 2.4	Knowledge Components shared across three different Applications	16
Figure 2.5	Objects and tree for a knowledge base to cross street	17
Figure 3.1	Portions of Nemati's knowledge warehouse architecture Focused in this research	24
Figure 3.2	Reference architecture of decision tree structure	25
Figure 3.3	Proposed framework for knowledge ETL	26
Figure 3.4:	Proposed Class structure for knowledge warehouse	30
Figure 4.1	Example of a general decision tree	33
Figure 4.2	An example of a binary decision tree	35
Figure 4.3	Regions for classification	36
Figure 4.4	Decision trees as classification algorithms	36
Figure 4.5	A binary Decision tree for Salary Decision	38
Figure 4.6	Decision Tree transformed for Warehouse	40
Figure 5.1	Relationship of variable	43
Figure 5.2	A decision table on how to spend a Saturday afternoon In spring	44
Figure 5.3	Rule probabilities for decision table	46
Figure 5.4	Decision table converted into a decision tree	46
Figure 5.5	Decision table format	48
Figure 5.6	Decision table for case study	54
Figure 5.7	Decision table transformed for warehouse	55

Figure 5.8	Simulation result of Decision table ETL algorithm	56
Figure 6.1	Make a call use-case represented as a tree	63
Figure 6.2	Use case for making a Phone call	65
Figure 6.3	Use case loaded in Warehouse	77
Figure 6.4	Simulation result of Use case ETL algorithm	68
Figure 7.1	Casual map of environmental forces and characteristics Of a technology	72
Figure 7.2	Knowledge map visualization for a news map	73
Figure 7.3	Knowledge source map	74
Figure 7.4	Process flow and knowledge flow map	74
Figure 7.5	Knowledge map of information sciences	76
Figure 7.6	Map of human knowledge	79
Figure 7.7	Partial knowledge map loaded in the warehouse	80
Figure 7.8	Simulation result of Knowledge Map ETL algorithm	81
Figure 8.1	Restaurant script from Schank & Abelson	87
Figure 8.2	Graphical representation of restaurant script after import in knowledge warehouse	90
Figure 9.1	Knowledge warehouse storage structure in the form of Directed graph	95

List of Tables

Table 5.1	Types of decision variables	43
Table 5.2	A Decision table for investment decision	45
Table 5.3	Rules made from the given decision table	54

Knowledge Extraction Framework from Scripts, Use-cases, Knowledge Maps, Decision Tables and Decision Trees for Knowledge Warehouse

By

**Bilal Ahmed
257-FAS/MSCS/F05**



A Thesis Submitted to the Faculty of

COMPUTER SCIENCE

in Partial Fulfillment of the Requirements of the Degree

MSCS

Fall 2005

International Islamic University

H-10, Islamabad, Pakistan

Chapter 1



Introduction

1.1 Motivations and Challenges

Decision Support Systems are now increasing their worth in the daily life cycle of organizations. In the early 1970s, Scott Morton defined Decision Support Systems as “*interactive computer based systems which help decision makers utilize data and models to solve unstructured problems*” [1]. Data warehousing being an integral part of these systems is flourishing rapidly. Data warehouses rely on data integrated from various sources to help decision making. Although the main reason of data warehouses is to get knowledge but currently data warehouses store and process data. They are a source of knowledge but they do not store or process knowledge directly. Knowledge management systems are also used for the same purpose of decision making. Instead of extracting knowledge from data as in the case of a data warehouse, these systems use knowledge base to provide decision support.

Now a need is being considered for a new generation of decision support systems by integrating knowledge management, decision support, artificial intelligence and data warehousing [5]. These knowledge-enabled systems will provide the infrastructure for capturing, cleansing, storing, organizing, leverage and disseminate not only data and information but knowledge as well[1].

This work tries to enhance the framework of Nematı for knowledge extraction and transformation for a knowledge warehouse on the same line as the concept of ETL in data warehouse. Knowledge can be extracted from many sources like scripts, frames, semantic nets, ECG, X-Ray, film clips, what-if cases, process modeling diagrams, mathematical models etc.

In this work, a framework for knowledge extraction and transformation for a knowledge warehouse is proposed. This framework is be used to device algorithms for knowledge extraction and transformation from five types of knowledge documents scripts, use cases, knowledge maps, decision tables and decision trees.

1.2 Background

Knowledge is an essential requirement for decision making. Knowledge, enables the decision makers to answer questions like “what to do?”, “when to do?” and “how to do?”. Data warehousing is used for providing the basic infrastructure for decision making by extracting, cleansing and storing huge amount of data. This data provides a solid factual foundation to analysts and decision makers and empowers them to succeed in a competitive business environment.

Knowledge management is a series of processes which include knowledge creation, organization, application and sharing etc. Ultimate goal of knowledge management is to provide knowledge to the right person at the right time for correct decision making [31].

Data warehouses are built for the sole purpose of providing knowledge but they do not have the capability to store or process knowledge directly. Instead of working on knowledge, they are designed to store and process data. They can be a source of knowledge but their storage and processing capabilities are limited to work on data only. Moreover only a fraction of knowledge is available in computers in a form extractable for a data warehouse. A vast majority of knowledge exists in other forms which are not suitable for loading in a data warehouse. Such as knowledge in the minds of employees in the forms of procedures, best practices, business rules, expert knowledge etc. A huge amount of knowledge required for decision making is in the form of tacit knowledge. This tacit knowledge may exist in the form of documents, audio and video libraries. Capabilities of data warehouse are limited and they can not store or process this form of knowledge. The question arises that if there are ways and means to store and process knowledge directly, why we don't use computer's capabilities to enhance the design of data warehouses to overcome these problems. Why not design a superset of data warehouses and enable it to store and process other types of knowledge.

A new generation of knowledge enabled systems by the name knowledge warehouse is proposed by Hamid R. Nemati [5] in his article “Knowledge warehouse: an

architectural integration of knowledge management, decision support, artificial intelligence and data warehousing". The writer has proposed a conceptual infrastructure to capture, cleans, store and organize knowledge for a knowledge warehouse.

Knowledge is available in a vast variety of forms. It may be tacit or explicit knowledge. A knowledge warehouse needs to store any type and form of knowledge. Even kneading the dough for bread is a form of knowledge; an ECG has knowledge in it. Decision tables and decision trees for diagnosis of a health problem or troubleshooting a vehicle engine also have knowledge. Even videos, sounds, user manuals etc have knowledge in them. Designing a storage structure to store and process any type of knowledge is a key issue for a knowledge warehouse.

Some work on a conceptual level for implementing a knowledge ware house is done by Hamid R. Nemati. He has proposed a theoretical framework of the knowledge warehouse. Anthony Dymond has suggested that the data warehouse model of Extract, Transform, Load (ETL) has a parallel in knowledge warehouse. As data is extracted, transformed and loaded in a data warehouse, a knowledge warehouse can be developed on the same principles by applying ETL on knowledge. Both nemati and Dymond have proposed in [5] and [2] respectively that an object oriented decision tree is a good storage mechanism for a knowledge warehouse.

This work proposes an object structure for storage and processing of knowledge in a knowledge warehouse. It also designs a framework for knowledge ETL operations as conceptualized by Dymond. Furthermore as knowledge is available in many different forms and transformation algorithms are required to load them in the proper form in a knowledge warehouse, five ETL algorithms is proposed, one each for five knowledge documents in my problem domain.

1.3 Research /Problem Domain

The problem addressed in this research thesis is to design a framework for Extraction and Transformation of knowledge to be accumulated in a knowledge warehouse.

Algorithms for extraction and transformation of knowledge from five types of knowledge documents scripts, use cases, knowledge maps, decision tables and decision trees are also proposed.

1.4 Proposed Approach

This research work is divided into two parts. First task is to propose a framework for knowledge ETL. It serves as a guideline structure for knowledge extraction and transformation from different sources to be accumulated in a composite repository in the form of useful knowledge. The second part of the thesis is the development of knowledge extraction algorithms for five knowledge documents i.e. Decision Trees, Decision tables, use cases, knowledge maps and scripts.

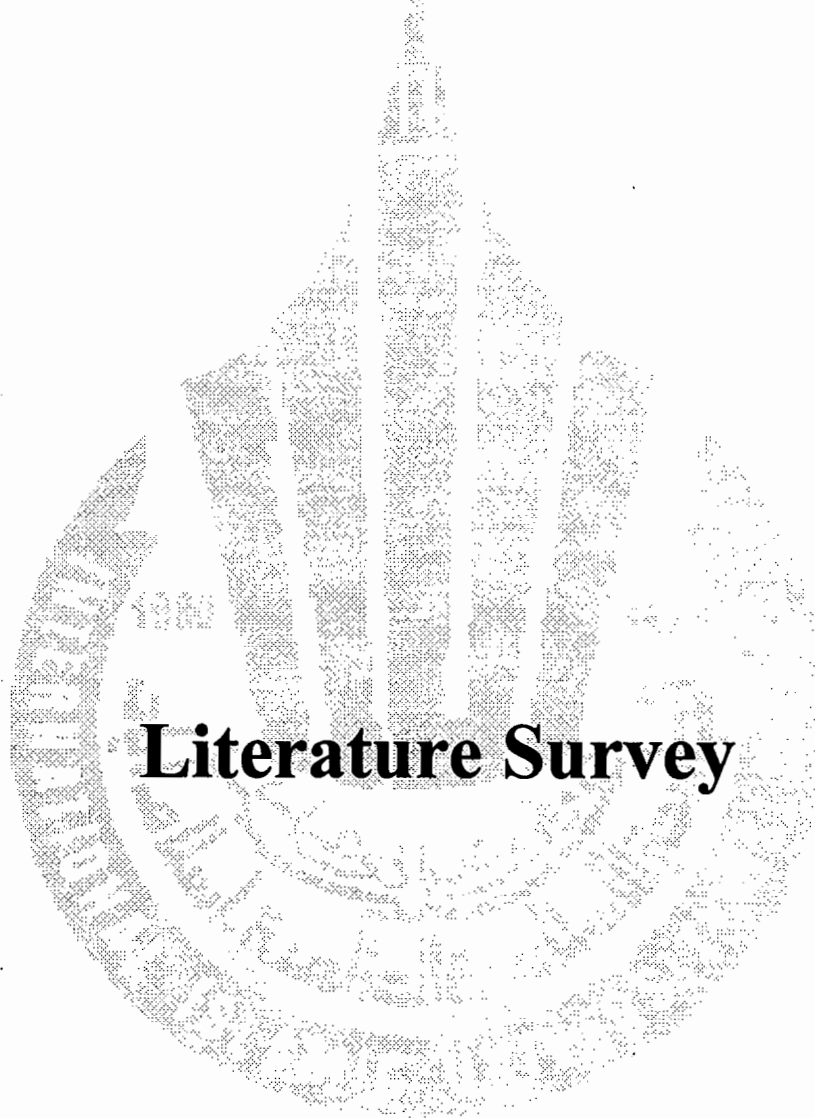
The base paper and other recent research articles surveyed, propose that an object oriented decision trees is better storage structure for a knowledge warehouse [5]. This work is carried out according to the guidelines set in the base papers but during the research work it was revealed that decision trees are not the best structure for this purpose. A better approach is mentioned in the future work section of this thesis.

1.5 Thesis Outline

This thesis consists of 6 major sections. First section covers the ETL framework of a knowledge warehouse. Remaining 5 sections consist of proposing ETL algorithms from 5 types of knowledge documents namely Decision Trees, Decision Tables, Knowledge maps, Use Cases and Scripts.

Each section consists of its own introduction and literature survey. Hence contribution of this article on the topic is also spread across the whole thesis.

Chapter 2



Literature Survey

2.1 Introduction

Decision Support Systems (DSS) are the systems intended to support managerial decision makers in semi structured decisions. The process of decision making is very complex and lots of work has been done to enable computers to assist in decision support.

All DSS use data, information and/or knowledge. These terms are mostly used interchangeably but factually they are different [1]. Here is a small definition of these terms for review

Data: Data items about things, events, activities and transactions are recorded classified and stored but they are not organized in the proper fashion to convey any specific meaning. Data items can be numeric, alphanumeric, figures, sounds and images [1]. Data is something that we attempt to gather and measure for example age, size or amount [10]. Data by themselves explain very little and it needs to be explained in the form of information.

Information: Information is data that have been organized so that they have meanings for the recipient. They confirm something the recipient knows, or may have surprise value by revealing something not known. We need to process data items so that the results are meaningful for an intended action or decision [1].

Knowledge: Knowledge consists of data items and/or information organization and processed to convey understanding, experience, accumulated learning and expertise which are applied to a current problem or activity [1].

In summary processed data is called information and processed information is called knowledge. This is the basic idea behind a data warehouse to produce knowledge. These warehouses capture store and process data to produce knowledge for decision making. A data warehouse provides an infrastructure that enables companies to extract, cleanse and store vast amount to corporate data from operation systems for efficient and accurate responses to user

queries [5]. This activity of business intelligence is depicted in [1, p130] as the following figure

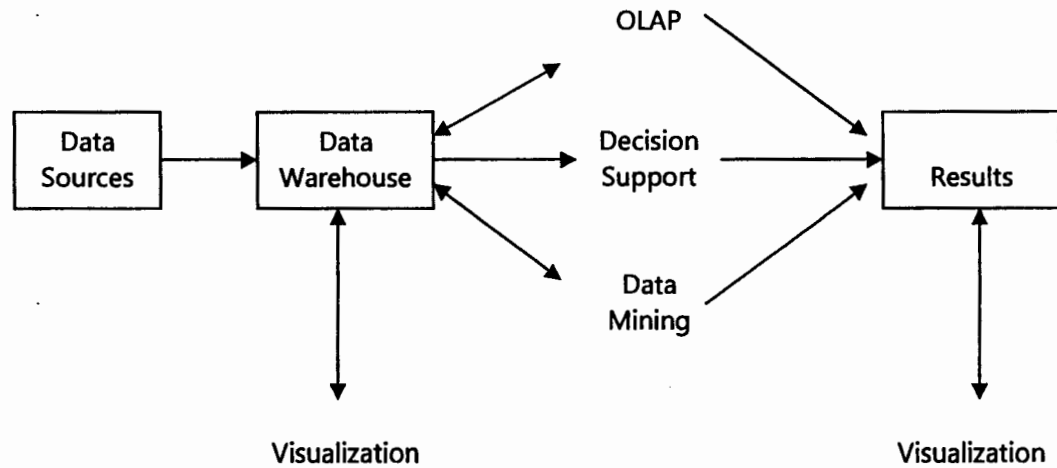


Figure 2.1: The Activities of Business Intelligence [1, p-130]

So the end product of all this activity is get knowledge. Companies spend time and money to capture important data and information about the market and their competitors and then process it to gain knowledge. Data warehouses have been an integral part of these decision support systems. Techniques like data mining are used to process the data to answer unstructured queries. But data warehousing and data mining are mid-game activities used to produce knowledge. End product is to apply the gained knowledge on company's business activity to produce some tangible results [2].

In real world only a fraction of knowledge required for business decision making can be stored in a data warehouse. A vast majority of knowledge is available in many other forms such as videos, sounds or documents. A large amount of knowledge exists in the minds of employees in the form of business rules, best practices etc.

Different types (tacit and explicit) and different forms (e.g. text streams, binary large objects, production rules, mathematical models, what-if cases) of knowledge need to be processed to develop an effective knowledge warehouse [5]. The knowledge warehouse also contains meta knowledge which can later be used to produce new knowledge throughout the organization.

Knowledge Management:

Knowledge management is defined by Nemati as “knowledge management is the practice of adding actionable value to information by capturing tacit knowledge and converting it to explicit knowledge; by filtering, storing, and retrieving and dissemination of explicit knowledge; and by creating and testing new knowledge” [5].

Technologies like DSS, computer sciences and Artificial intelligence can all be used in knowledge management. These technologies can store knowledge, process and enhance knowledge, can improve knowledge sharing in the organization, and can convert tacit knowledge to explicit knowledge.

A four step process termed as “The Knowledge Spiral” for creation of new knowledge for knowledge management is given in the picture below. This knowledge spiral and all its phases are discussed in detail by Nemati in [5] and are very useful for understanding the knowledge management process. It shows not only the phases of knowledge transformation but also discusses different techniques useful at any stage and different data types which can be used to store knowledge at each stage in the knowledge spiral.

The four main processes identified are Externalization, Socialization, Internalization and Combination

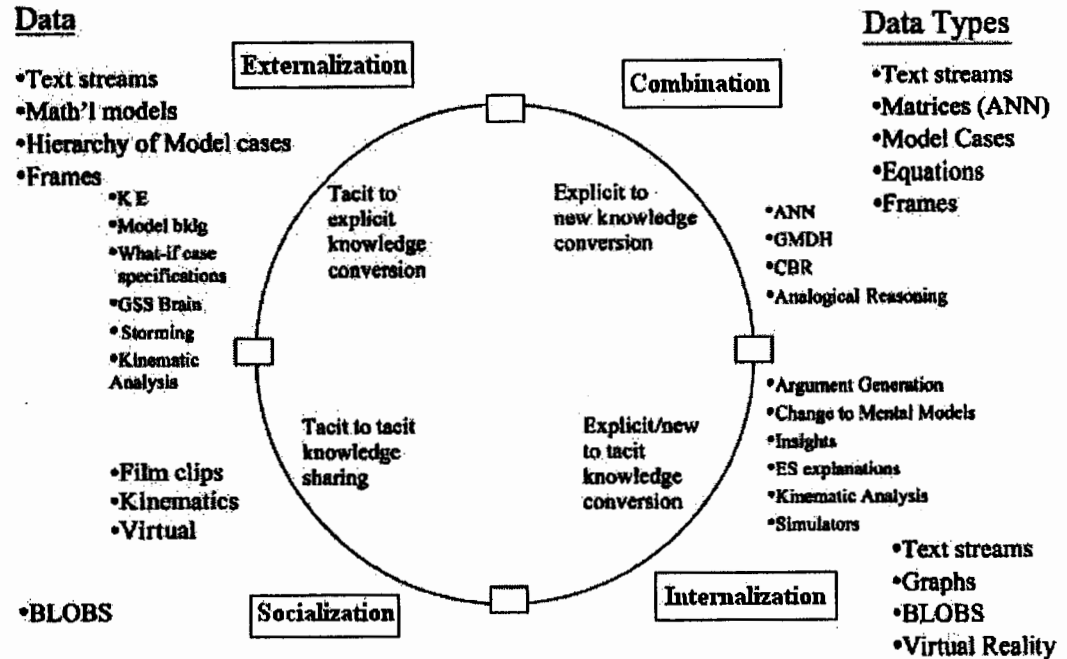


Figure 2.2: Knowledge Spiral [5]

The knowledge spiral consists of 4 steps.

Socialization: This is the process of sharing tacit knowledge in any form like technical skills, experiences [5]. For example a craftsman learns the art from its master. The knowledge in this case is transferred from one mind to the other without being codified in any form.

When discussing in the context of computer sciences, we can share tacit knowledge by creating digitized films or animations of the process and make it available for searching. This knowledge can be better used by adding commentary of knowledge workers who have the tacit knowledge about the process. Facilities like slow motion and zooming can be provided to share the knowledge in the proper way.

Externalization: This is the process of converting tacit knowledge to explicit knowledge. It is also called articulation [5]. The knowledge is codified into some useful form which can be processed by the computer.

Although this process seems confusing as tacit knowledge by definition is the knowledge which cannot be stored in a form to be processed by the computer. But techniques and guidance are available in literature for this process.

Different techniques are employed to obtain knowledge from knowledge workers in verbal or animation or any other form and then interpreting that knowledge to infer the underlying knowledge and reasoning process.

Combination: Combination or integration is the process of combining different types of knowledge stored in process-able form to generate new knowledge. New patterns and relations are discovered from existing knowledge.

The knowledge converted to computer readable form in the externalization process can now be process and analyzed in the context of already available knowledge to create new explicit knowledge.

Internalization : Internalization is the process of testing and validating the newly discovered knowledge in the proper context hence generating new tacit knowledge [5].

Computers can be very useful at this stage as they can provide valuable aid for the knowledge workers to learn.

Goals of making a knowledge Warehouse: The main goal of a knowledge warehouse is to provide an intelligent analysis platform to the decision makers that can be helpful in all stages of knowledge management[1]. A knowledge warehouse must be able to store, retrieve and process knowledge in many forms. It must also be able to perform analysis tasks to generate new knowledge.

which the key issues for the development and implementation of a knowledge warehouse are mentioned. Some of these factors are :

1. Requires a lot of time for implementation
2. Cost of development may become out of budget for many organizations
3. Support from top executives
4. Users involvement and participation
5. Well-defined business objective and goals
6. Resources adequacy issues.
7. Organization and political issues within the organization
8. Technological issues
9. Process management issues
10. Plans and communication issues
11. Values and ethics issues
12. External to organization issues

All these issues may determine the success of a DSS implementation and these also apply for a knowledge warehouse. However in case of knowledge warehouse, some more factors come into play. As the main focus of knowledge warehouse is to tap the intellectual capital of the organization and tacit knowledge in the minds of knowledge workers the following factors also become important

1. The technical knowledge management infrastructure
2. Building a culture in the organization for sharing the knowledge
3. Facilitation of knowledge oriented connections, coordination and communication.

Nemati has given a list of activities for the successful development of knowledge warehouse architecture in an organization.

-
1. Designing and implementing the techniques to identify and record both knowledge and ignorance. Designing the processes to share, use and protect the knowledge.
 2. Creating a knowledge sharing environment in the organization and encouraging employees to share their knowledge.
 3. Articulating and communicating the purpose and nature of knowledge management and connecting it to the strategies and operational initiatives of the organization [5].

Micheal Yacci proposed another use of knowledge warehouse in the context of reusing knowledge components. The fact highlighted in his research is that most of the knowledge components are developed for a particular purpose and knowledge is hardcoded in the material by trainers, performance support, documentation groups etc. this makes the knowledge non-reusable[10]. Such non-reusable softwares are termed as “brittlewares”[10] . It is suggested that such material must be revised in an organization to keep it up to date.

In his research, Yacci has proposed that a knowledge warehouse can be a solution to this problem and can enhance re-usability of knowledge documents. He defined knowledge warehouse as an information repository in which knowledge components are cataloged and stored for reuse [10]. Such a warehouse can provide different views of the same knowledge. He suggests that this warehouse can be used for ad-hoc queries such as electronic performance support systems, intelligent help, or reference material [10]. He has proposed an abstract structure of the knowledge warehouse reproduced in the figure below:

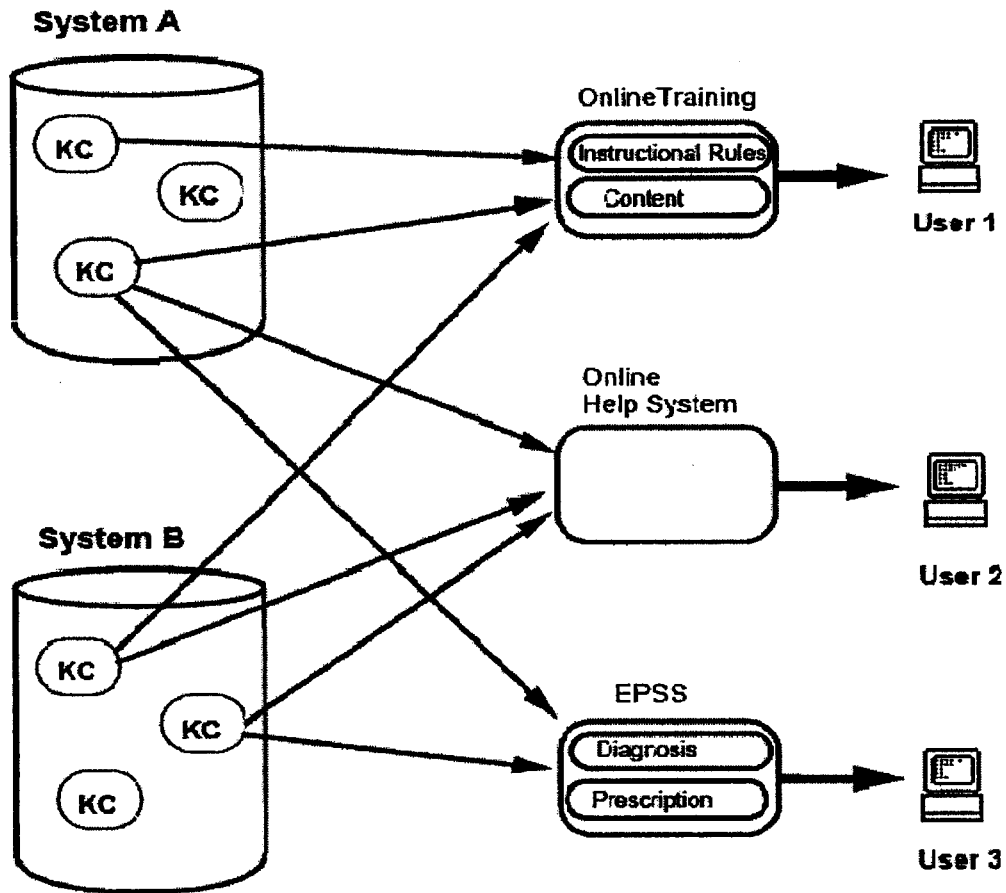


Figure 2.4: knowledge components shared across three different applications [10,p4]

He proposed that knowledge can be stored in the form of “Knowledge Components (KCs)” to be stored at different physical locations. System A and System B in the figure show different physical storage locations. Different users can use different knowledge components in different views. Three users are using three different applications giving those views of the required knowledge components.

2.2 Related Research/Technologies

As a major portion of this thesis is the knowledge extraction algorithms, hence the related part to this thesis is the storage mechanism of a knowledge warehouse.

Nearly all the literature surveyed during the study was of the view that the storage mechanism of a knowledge warehouse needs to be an object oriented knowledge base.

Storage module is a primary component of a knowledge warehouse. This knowledge base will be able to handle a wide variety of knowledge objects like data streams, relations, movie clips, animations, sounds etc. meta knowledge handling capability is also very important.

Main advantages of storing the knowledge in object form are

1. Meta knowledge can be stored with the knowledge. Knowledge source information, its analysis techniques can be stored together in an object providing encapsulation.
2. Executable routines for processing the knowledge can also be stored in the same object.
3. Object oriented concepts like inheritance, polymorphism, method overloading etc can be used to efficiently store and process the knowledge in the form of super-class and sub-class relationship.

Anthony Dymond goes one step ahead by proposing that an object oriented decision tree is the best form to store knowledge in a knowledge warehouse. He described a small example of a decision to cross the street in the form of an object oriented decision tree. The objects and tree as proposed is given in the picture below.

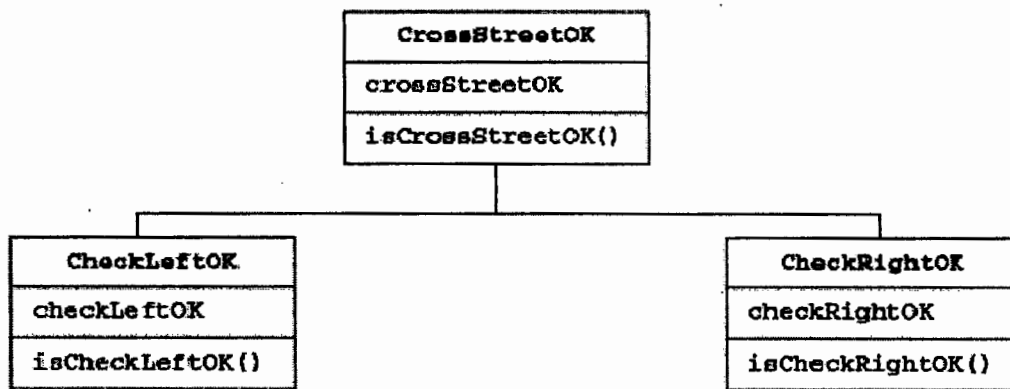


Figure 2.5: Objects and tree for a knowledge base to cross the street [2]

He proposed that the methods will be activated by the way of tree search algorithms these algorithms are procedures to traverse the branches of tree [2]. The code for CrossStreetOK object will be implemented as

```
If CheckLeftOK = 'True' and CheckRightOK ='True' then
```

```
    CrossStreetOK = 'True'
```

```
End if
```

An algorithm touches the nodes and runs the methods as the “focus of attention” of the algorithm. For example, a depth first search algorithm will start at the root node and traverse the tree in the following order.

1. CrossStreetOK
2. CheckLeftOK
3. CrossStreetOK
4. CheckRightOK
5. CrossStreetOK

According to Dymond, this type of tree search supports a control structure. Assume that when the search reaches the node CheckLeftOK and it finds CheckLeftOK.CheckLeftOK = 'False'. This result is passed to CrossStreetOK function, where CrossStreetOK.CrossStreetOK can now be set to False. The search process can be stopped here as it is no longer necessary to check CheckRightOK. This type of executable code that will set the value of decision variable can be a nice way to implement this decision making [2].

Executable code can also be very useful for performing some other tasks like taking an input from user at any stage of decision making. This technique has been used in this work during implementation of algorithms. This approach of using a decision variable and a decision function is found very helpful in this research.

2.3 Limitations

Knowledge warehouse is still a theoretical concept. The work done is in great detail but it is far from implementation. Nemati has proposed architecture for knowledge management and his work can be considered a milestone in the history of decision support systems but all his work is theoretical. Only an outline is proposed for the new concept which can be explored in further detail. In his paper he admits that developing even a prototype for the proposed knowledge warehouse is beyond the scope of his paper.

Dymond has proposed that knowledge warehouse has a parallel in ETL just like a data warehouse. He has also proposed the knowledge storage structure for the warehouse implementation. His work is on a very basic level and needs to be refined and explored further.

2.4 Summary of Chapter 2

Knowledge is a key asset for organization growth. Companies are investing on data warehouses as Decision support systems to provide them with the necessary knowledge about market and their competitors.

Data warehouses have a built in limitation as they only store and process only data. The expected output of such decision support systems is knowledge but they do not process or store knowledge directly. Moreover most of the knowledge of an organization is in the minds of its employees in the form of tacit knowledge. This knowledge is never documented or shared in an organization and hence can never be processed to generate more knowledge.

A new dimension of decision support systems is proposed by H.R. Nemati to enable DSS to store and process knowledge to form a knowledge warehouse. This type of warehouse can store any type of knowledge and share it within and outside of the organization.

Dymond has proposed that an ETL operation has a parallel for knowledge warehouse on the same grounds as a data warehouse.

All the research work is on abstract level and needs to be further explored to make this field a reality in the years to come. Knowledge warehousing has the potential to become a new trend in the field of decision support systems.

Chapter 3



Requirement Analysis and System Design

3.1 Introduction

The problem domain for this research falls in the category of decision support systems. An emerging concept in the field of decision support systems is knowledge warehousing which is considered to be the next step after data warehousing for decision support systems.

Some conceptual level work has been done to explore utilization of this field in decision support systems. Nemati has given a detail conceptual model of knowledge warehousing in his article [5]. The work done is very detailed and it explores the various concepts of a knowledge warehouse but still it is a conceptual level work and the field needs to be explored further in detail to make this concept a reality.

3.2 Problem Scenarios

Although the foundation of knowledge warehouse is put firmly in place by the research work of Nemati but all the concepts are in abstract form. To collect and store knowledge from different sources to create a giant warehouse containing entire knowledge of the organization is a tedious task. An extract transform and load mechanism is required to collect knowledge from different sources available in different forms to load it in a consistent form in the warehouse.

A frame for knowledge ETL from knowledge sources on the same grounds as a data warehouse is required to serve as a guideline for importing knowledge into the warehouse.

Furthermore, knowledge extraction and transformation algorithms are required for this ETL operation. Knowledge is available in many different sources in an organization ranging from audio speeches, video documentaries, ECG, semantic nets, frames, use-cases and a lot more.

An algorithm is required for each and every knowledge source to import the data into the warehouse. These algorithms will serve as the guideline for knowledge ETL operations.

3.3 Focus of Research

This research explores the area of ETL for a knowledge warehouse on the same grounds as ETL for a data warehouse. A conceptual level framework for knowledge ETL operations is proposed.

Furthermore physical storage structure of a knowledge warehouse is also proposed. This structure enhances the knowledge class structure of Dymond and includes attributes for better navigation of the knowledge tree.

Furthermore knowledge extraction from 5 types of knowledge documents i.e. Decision Trees, Decision Tables, Use cases, Knowledge maps and scripts will be discussed in detail and knowledge extraction algorithms for loading knowledge extracted from these sources into a knowledge warehouse will be proposed.

3.4 Design Requirements

The framework is required to meet the following requirements to serve its purposes

- It must cover all the areas of ETL operations. Knowledge ETL is different from data ETL.
- It must be able to cater for both types of Knowledge Tacit and explicit.
- It must also cover meta-information required about each knowledge source for the transformation of knowledge properly.

3.5 Reference Architecture

This is first effort to device a framework for knowledge ETL. It takes reference of the knowledge warehouse architecture given by nemati in [5]. The architecture given by nemati in [5] is shown in the figure below and the portion of this architecture covered in this thesis is highlighted.

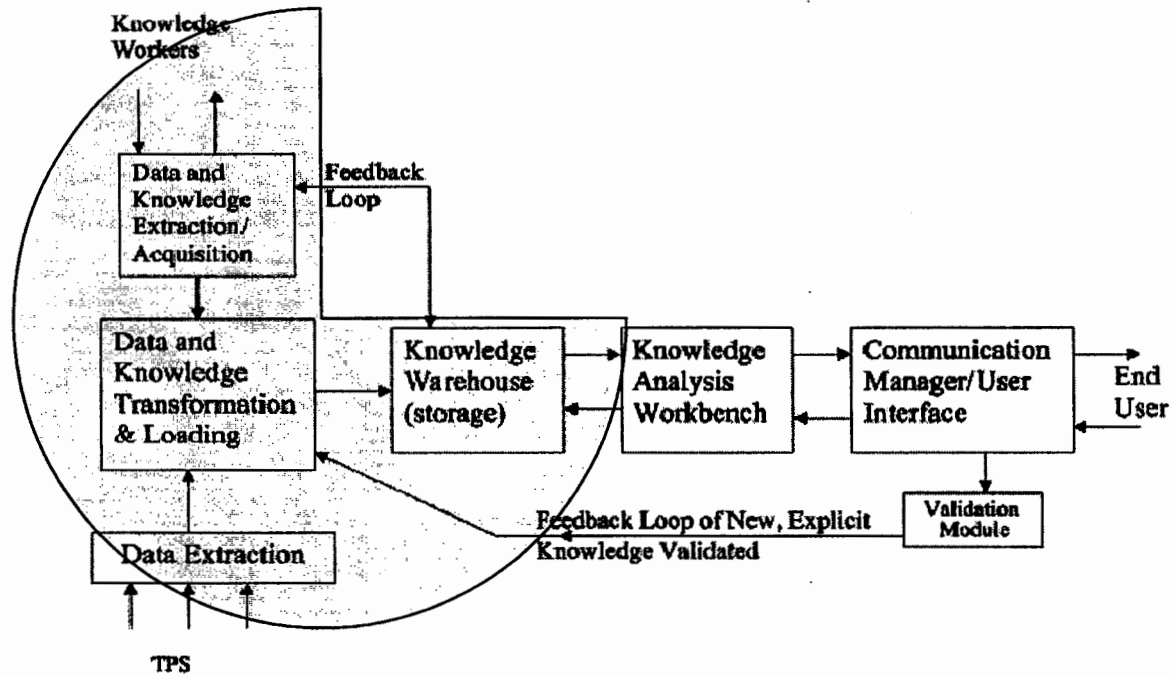


Figure 3.1: Portion of the Nemati KW architecture [5] focused in this research

Only the part related to knowledge extraction, its transformation and loading is covered in this thesis. The feedback loops of the architecture relevant to generating new knowledge and storing it in the warehouse are not included.

A node structure for the knowledge warehouse decision tree node is also proposed in this thesis. This work is based on the Dymond structure of knowledge decision tree given in [2]. The structure presented by Dymond is a primitive level structure proposing the use of a decision variable and a decision function. This structure provides a very good baseline as it uses object oriented approach. A proper class structure is used to achieve to take advantage of object oriented design such as encapsulations, inheritance etc.

The structure is shown in the figure below with a brief description of its components.

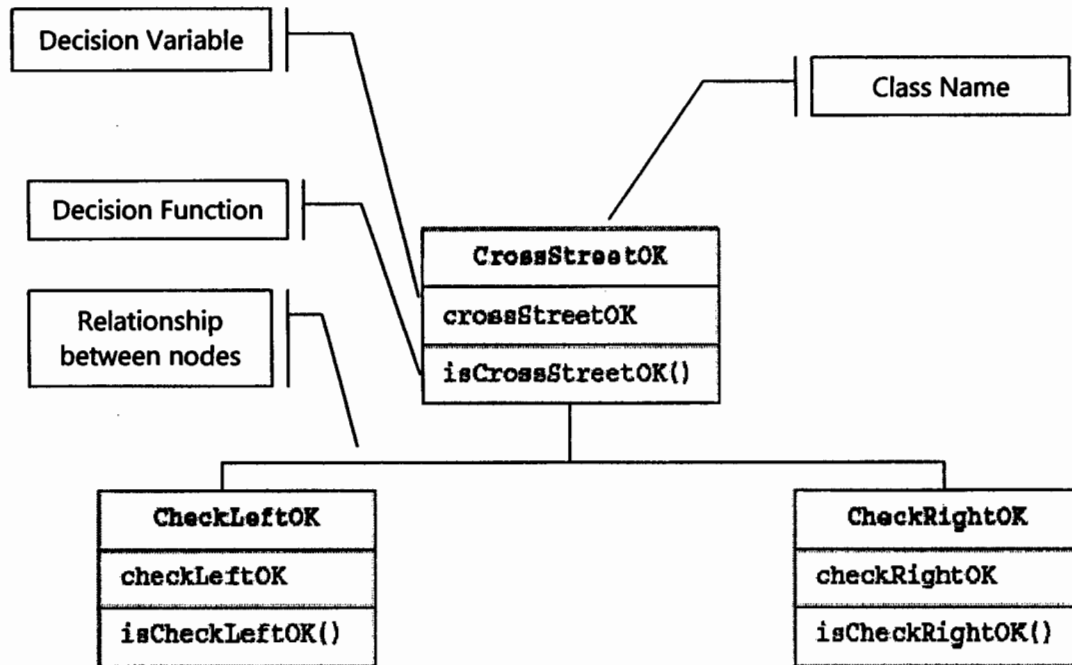


Figure 3.2: reference architecture of Decision tree structure [2]

This reference structure is enhanced by adding fields for traversals and storing tacit knowledge in the same node.

3.6 Proposed ETL Framework for knowledge warehouse

Based on the knowledge warehouse architecture of Nemati, a framework knowledge ETL operation is proposed in this thesis. This framework is designed to cater for all the requirements that may come across in the knowledge ETL operations. It can handle both tacit and explicit types of knowledge.

The ETL framework proposed for knowledge extraction and transformation is given in the figure below.

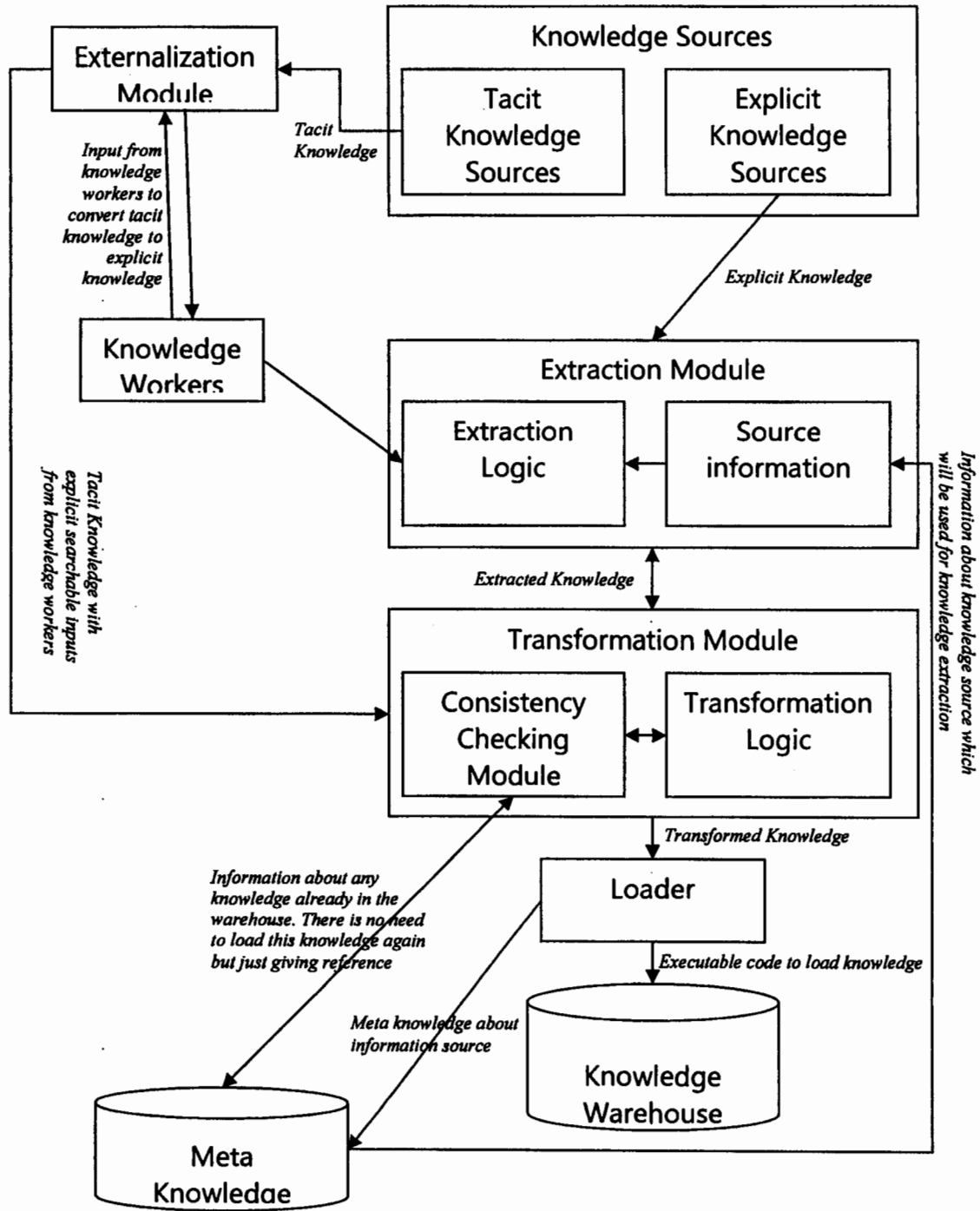


Figure 3.3: Proposed framework for knowledge ETL

A detailed description of each of the above components is given below.

Knowledge Workers: All the organizations employ some workers due to their knowledge in a subject matter. These workers are the intellectual asset of the organization. Workers who have mastered a certain skill for example lawyers,

computer experts, scientists, banking experts can be called knowledge workers. This is the academically trained workforce of an organization.

This term was invented by Peter Druker in 1959 for a worker who works with information or the worker who develops the use of knowledge in the organization [6].

These workers have the necessary knowledge in their minds which must be put in a documented shape in a knowledge warehouse. A knowledge warehouse must be able to organize the knowledge from these knowledge workers and share them inside the organization.

Knowledge Sources: Knowledge warehouse is all about collection of knowledge. Just like data is captured from many sources for a data warehouse, knowledge is collected from different sources for a knowledge warehouse. These knowledge sources can be in any form tacit or explicit.

Externalization Module: Externalization is the process of converting tacit knowledge to explicit knowledge. As tacit knowledge by definition is not in a form to be processed by computers, there is a need to externalize this knowledge into a form so that it can be stored in the knowledge warehouse.

All the knowledge in tacit form, either from any tacit knowledge source or described by a knowledge worker must go through the externalization phase to make it compatible with a form process able and searchable by the computer.

For example, a vast variety of knowledge can be collected in the form of videos. But videos are not searchable or process able by computers to extract knowledge from them. We may forward this video to a knowledge worker to add searchable commentary to it. The knowledge worker may add searchable keyword to the video so that it can be searched later. Knowledge workers can even categorize these videos by author, language etc.

Extraction module: This is the module responsible for extracting knowledge from different sources so that it can be processed for loading into the warehousing. Knowledge in different sources can be poorly structured and needs to be put into proper shape before running the transformation procedure. Some meta knowledge about source and format of knowledge may also be incorporated at this stage.

As knowledge is stored in complex forms and not simply in relational tables, hence extraction of knowledge is much more complex than extraction of data. For this reason the extraction module will need two sub-components for proper extraction i.e. source information and extraction logic module.

A knowledge document can have many types of knowledge which needs to be tackled differently and stored at different places in the warehouse.

Source Information: This module will have all the knowledge about the knowledge source. It will provide information about which knowledge is stored where in the source and what its type is. For example knowledge available on a news website may contain images and textual data available at many places in the web page. All that needs to be extracted and put in proper shape so that it can be linked later to retrieve knowledge later.

Warehouse manager has to provide the knowledge extraction logic to the system as how to extract specific knowledge from the source. It needs to make sure that the knowledge extracted is semantically and contextually correct. Extraction module will have this extraction logic provided by the knowledge managers for each source.

Transformation Module: This module will transform the extracted knowledge and re-structure it according to the storage structure of the knowledge warehouse. As the knowledge warehouse will store the objects in object oriented form hence there is a need to transform the information retrieved into object oriented form. This will require creating objects, writing executable routines, storing Meta knowledge in the proper form, coding analysis routines etc. This module will have the “consistency checking module” and “transformation logic module” as its sub-modules.

As warehouses retrieve information periodically from the same source there is a need for “**Consistency Checking module**” to ensure that the knowledge from same source are linked together and the nodes are created in proper relationship to each other. This module will take input from meta-knowledge where information about sources is available in the proper form. Sometimes there might be a need to store the retrieved information in an existing node of the warehouse.

Transformation logic module will have all the source specific logic to transform the knowledge retrieved in to the warehouse storage structure. This logic will be provided for each source by the warehouse manager.

Loader: This module will physically load the knowledge and meta knowledge into the warehouse. This module will physically create the objects, link them in an object-oriented hierarchy, create persistent storage variables and write the executable code for each object.

3.7 Proposed Node Structure of a Knowledge warehouse

Physical storage of the knowledge is the key factor in the implementation of a knowledge warehouse and it was a basic requirement for developing algorithms for knowledge transformation.

None of literature surveyed proposed a node structure for the warehouse. Researchers have proposed that it needs to be in the form of an object oriented tree but the node structure was not provided anywhere. This work also proposes a knowledge tree node structure for the knowledge warehouse. the proposed node structure is shown in the figure below:

TH 7635

Class KWNode
Decision Attribute
Node Type
Blob object
Text Description
Array of Affirmative Child Node
Pointer to Negative Child Node
Pointer to Parent Node
Relation with parent
Decision Function
Analysis function

Figure 3.4: Proposed Class structure for knowledge warehouse

A brief description of these attributes is as follows.

Decision Attribute: This variable will be used to return the status of decision function. In most of the cases it will be a Boolean value. This variable will be set in the decision function or analysis function to communicate the result of function. In section 2.2 the use of this variable can be seen as example.

Decision Function: Every node needs an executable code to perform the decision functionality and then set the decision attribute accordingly to return the result. CheckRightOk and CheckLeftOk are examples of decision function given n section 2.2.

Analysis function: To improve storage efficiency we can also store the analysis functions in the same class. Keeping the storage structure object oriented can also be very helpful. Mostly the analysis tasks on the inherited classes will vary slightly from

that of the base class. This can make analysis code very well managed and storage efficient. Furthermore method overloading and polymorphism can also be very helpful.

Node Type: All the knowledge from any source will be stored in this same node structure. This node may contain text data, numerical data, video clips, audio clips etc. we can use this variable to get information about the type of data stored in the node. Source of knowledge is deliberately missed here as it will be stored in meta knowledge.

Blob Object: A blob type variable to store any blob field. For example images, videos, audios etc.

Text Description: This field will be used for tacit knowledge such as videos and audios. As it is difficult for computers to search tacit knowledge blob field, hence a text commentary by some knowledge worker in searchable form can be stored with each blob. This may include the keywords for knowledge searching.

Array of Affirmative child node: For majority of the cases, it will be a binary tree hence every node will have 2 nodes. This field will have a pointer to the child node to follow when decision attribute has the value "True". It is an array just to accommodate decision trees that are not binary decision trees.

Pointer to negative child node: This field will have a pointer to the child node to follow when decision attribute has the value "False".

Pointer to parent node: This field will have a pointer to the parent node. each node will have a pointer to the parent node. For the root node it will be NULL.

Relation with parent: This field will save the value that this node is affirmative child node of the parent node or negative child node. It can be very helpful while backtracking for a decision making or some analysis requirements.

3.8 Summary of Chapter 3 (Requirement Analysis and System Design)

This chapter focuses on the knowledge ETL framework and the decision tree node structure proposed in this thesis. The work done in this thesis does not end here and five knowledge extraction algorithms will be proposed in the next 5 chapters, one for each type of knowledge document.

CHAPTER 4



Knowledge Extraction Algorithm for Decision Trees

4.1 Introduction

A decision tree can be regarded as a deterministic algorithm for deciding which variable to test next based on the previously tested variables and the results of their evaluation, until the function's value can be determined [5].

Decision trees are sequential evaluation procedures and are often used for system analysis and decision theory [1]. Just like other sequential operational procedures, decision trees have widespread applications in database, decision table programming, concrete complexity theory, switching theory, pattern recognition and taxonomy [5]. Nodes in a decision trees represent goals and links represent decision [1]. Value of a variable is determined at each node and the next node is decided accordingly. Next node may evaluate the value or output the value.

4.2 Literature Review

Definitions of Decision trees:

Definition 1: Decision trees are rooted, ordered, vertex-labeled trees where each node has either m_i children for some i , $1 \leq i \leq n$, or none (leaf) [5].

Moret's Definition : Let $f(x_1, x_2, \dots, x_n)$ be a function of discrete variables. If f is constant or null, then the decision tree for f is composed of a single leaf labeled by the constant value or by the null symbol. Otherwise, for each x_i , $1 \leq i \leq n$, such that at least two restrictions, say $f_{|x_i=k_1}$ and $f_{|x_i=k_2}$, are not null, f has one or more decision trees composed of a root labeled x_i , and m_i subtrees, which are decision trees corresponding to the restrictions $f_{|x_i=0}, \dots, f_{|x_i=m_i-1}$, in that order. [5].

Examples of uses of a Decision Tree

An example of a decision from [1] to diagnose a car starting problem is given in the figure below.

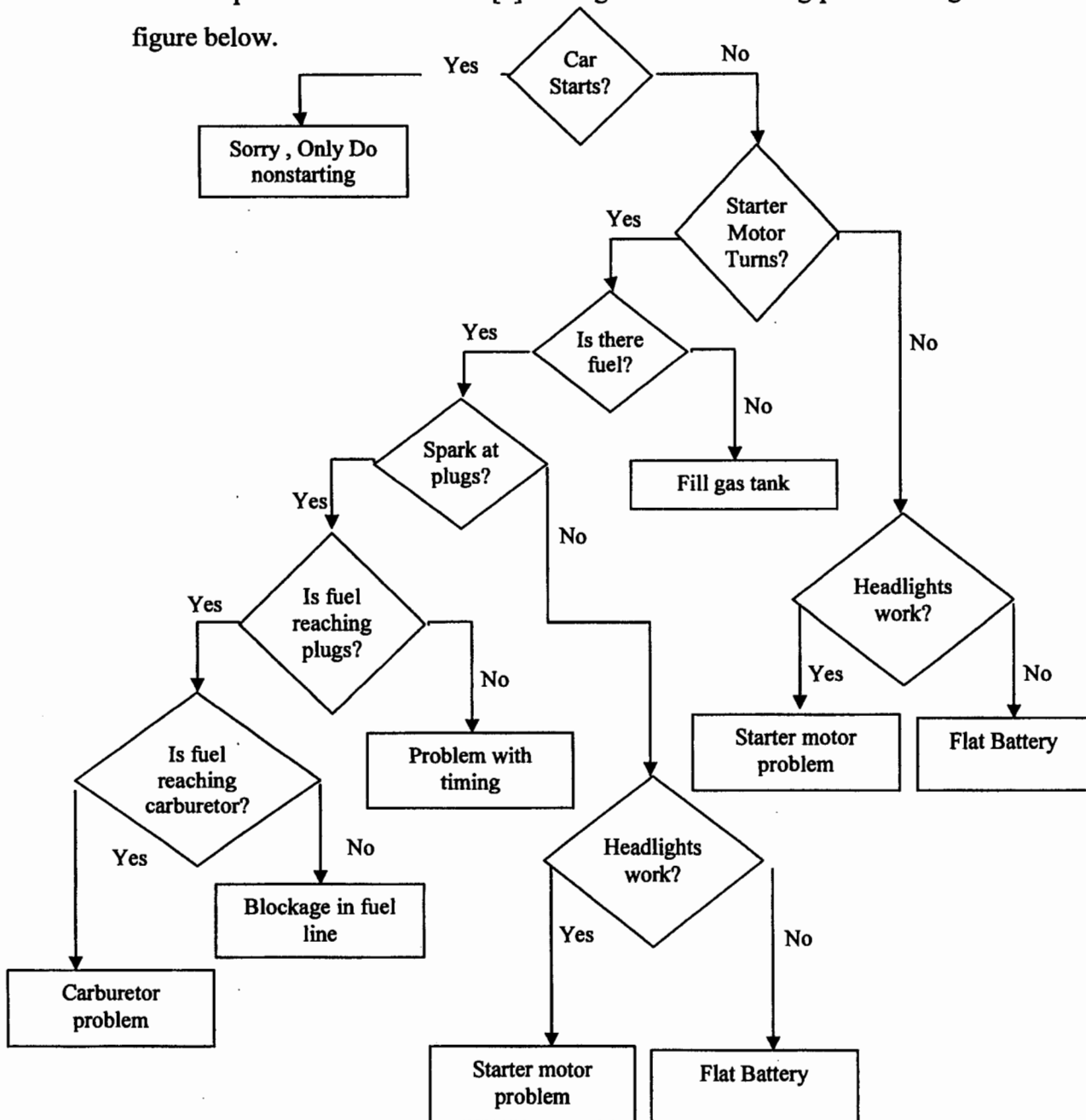
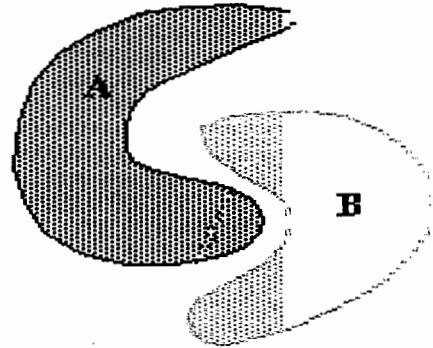


Figure 4.2 : An example of a binary decision tree [1]

Each node in the tree is for a decision. For example the root node is a question that will have a Boolean answer either in the form of “Yes” or “No”. Each node has two branches, one for each of the possible answer. Each branch of a decision tree will either lead to a new question or a leaf node which tells the diagnoses result.

It is clear from the example that decision trees contain knowledge in them. They are made by knowledge workers to help in decision making.

Decision trees can also be used for region classification. An example of decision tree usage for region classification from [32] is given below.



Consider the regions A and B in the figure 5.3.

Imagine these regions are placed in an $[x, y]$ plane. Many algorithms can be found in literature to classify the regions in $[x, y]$ planes. Input of our algorithm in x and y coordinates of a point, the algorithm is required to tell the region in which the point lies.

Figure 4.3: Regions for Classification

Decision trees can be used as classification algorithms for such problems. The decision tree to solve this region classification problem given in [32] is given below.

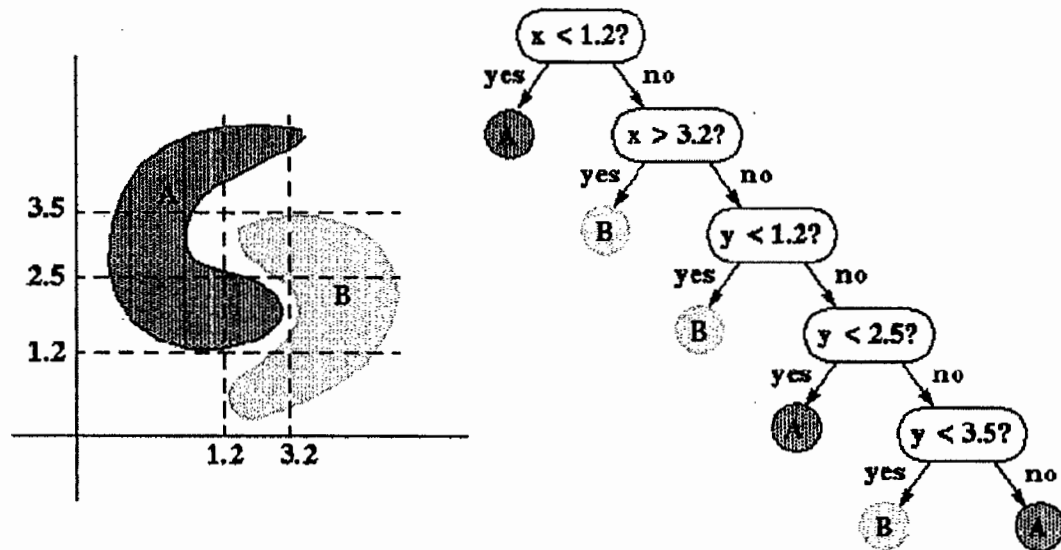


Figure 4.4: Decision trees as classification algorithms [32]

4.3 Problem Scenario for the Decision Tree ETL Algorithm:

As discussed in the previous chapters, knowledge needs to be extracted and transformed from knowledge documents before storing it in a knowledge warehouse. Decision trees have widespread applications and have been used to document sequential evaluation procedures.

We can extract knowledge from decision trees and transform and store them in a knowledge warehouse to use it for decision making.

For this research the storage structure of the knowledge warehouse is object oriented decision trees. The structure of the object and other details about storage has been discussed in the previous chapters.

The transformation algorithm for storing knowledge is proposed below. This algorithm will take input a decision tree and make necessary transformations to store the knowledge in a knowledge warehouse in accordance with the storage structure proposed earlier in this thesis.

4.4 The Proposed Algorithm for Knowledge extraction from Decision Trees:

\\Comment: this function will be called for the root node of input tree and it will use recursion to formulate the complete tree

```
CreateNode(Node InputNode, Is_Root Boolean, Affirmative Boolean)
{
    If Node is NULL then Return
    n = new KWNode
    n.nodetype = "Decision"
    n.createDecisionFuntion( Node.decisionText, By Ref n.DecisionAttribute)

    if Is_Root then
        n.parentNode = NULL
        RelationWithParent = NULL
```

```
else
```

```
    n.parentNode = Node.ParentNode
```

```
    RelationWithParent = Affirmative
```

```
End if
```

```
n.AffirmativeChildNode = CreateNode(affirmativeNode,False,True)
```

```
n.NegativeChildNode = CreateNode(NegativeNode,False,False)
```

```
}
```

4.5 Case Study for the Proposed Decision Tree ETL Algorithm

A case study approach is used in this section to validate and test the proposed algorithm. This will take a sample decision tree to apply the proposed algorithm and will graphically represent the output which is in the form of proposed physical storage structure of the knowledge warehouse. It will not only validate the algorithm but will represent the final output of the knowledge warehouse storage structure proposed in this thesis

Consider the binary decision tree in the figure below for making a decision of salary calculations of salaried and hourly workers

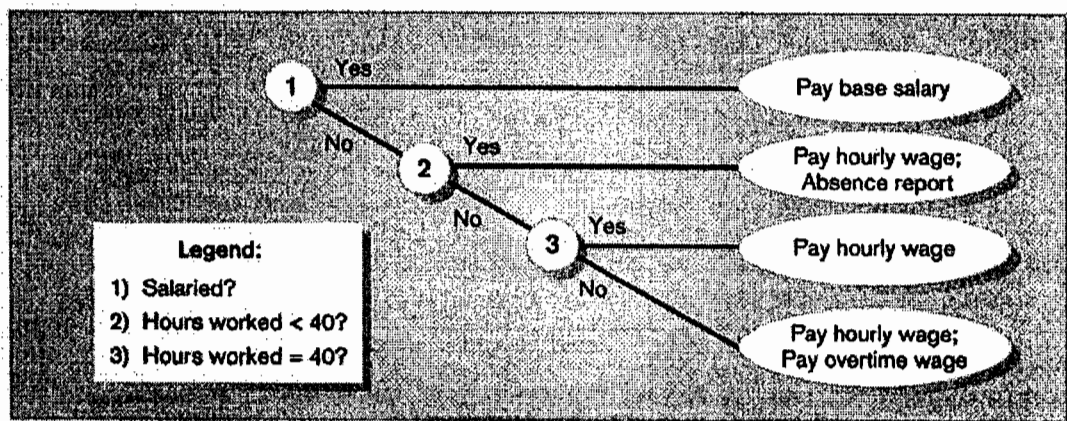


Figure 4.5: A binary Decision tree for Salary Decision [33]

The proposed algorithm for performing ETL operation on decision trees will be applied on the decision tree given in the above figure. Application of the algorithm will transform the decision tree into the proposed knowledge warehouse storage structure.

A decision for salaried workers is made at first node. Base salary is paid to the salaried workers. If the worker is not a salaried worker, then an hourly rate applies which is decided in the preceding nodes.

At second node a decision is made that the hours worked by the worker are less than 40 hours. An affirmative answer shows that the worker has been absent from duty. Hence an hourly rate is applied and absent hours are accounted for. A negative answer at node 2 leads to node 3 which checks that has the worker worked for 40 hours? An affirmative answer to this question leads to a plain hourly rate. A negative answer implies that the worker has worked for more than 40 hours and a decision is made to pay not only the hourly wage but overtime as well.

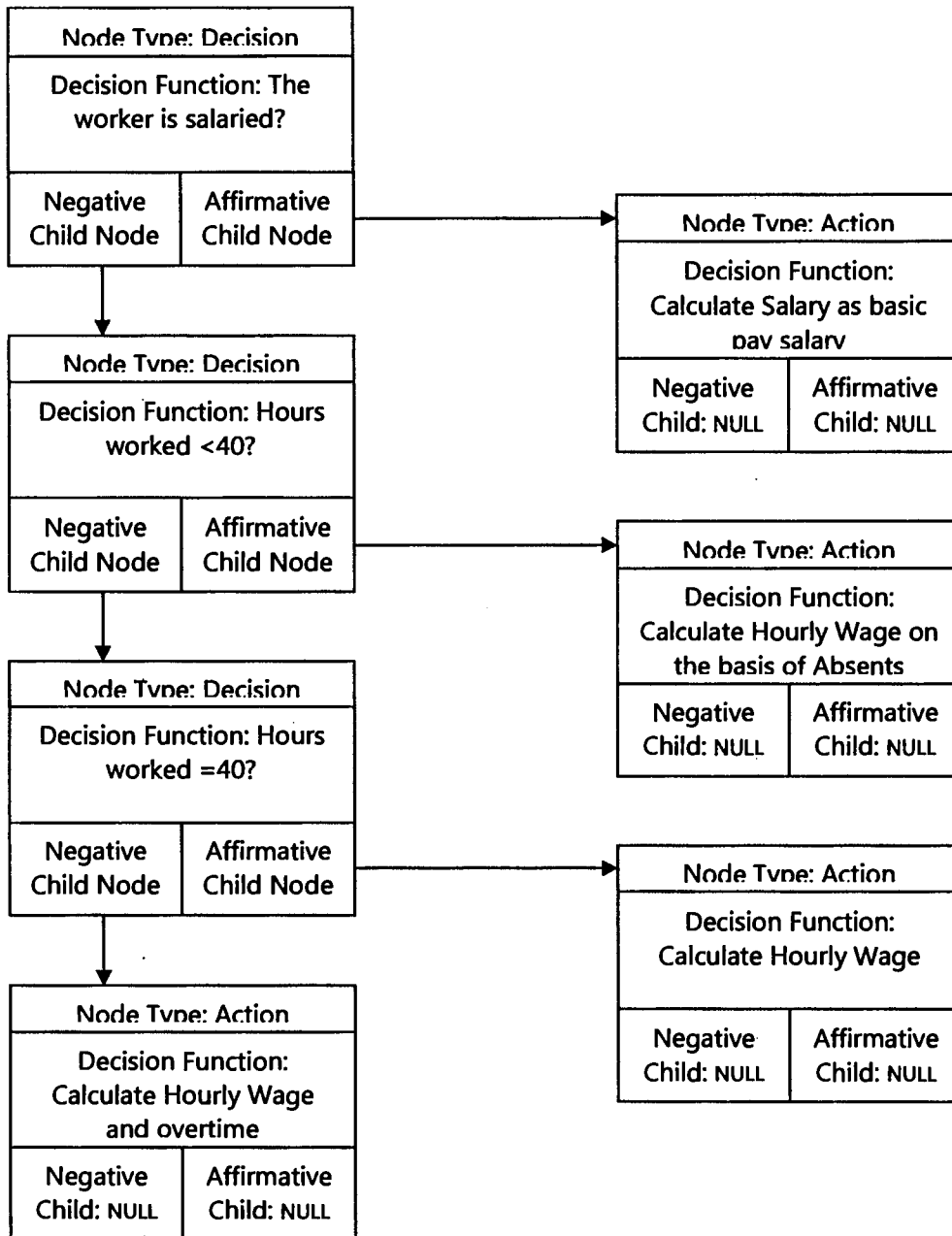
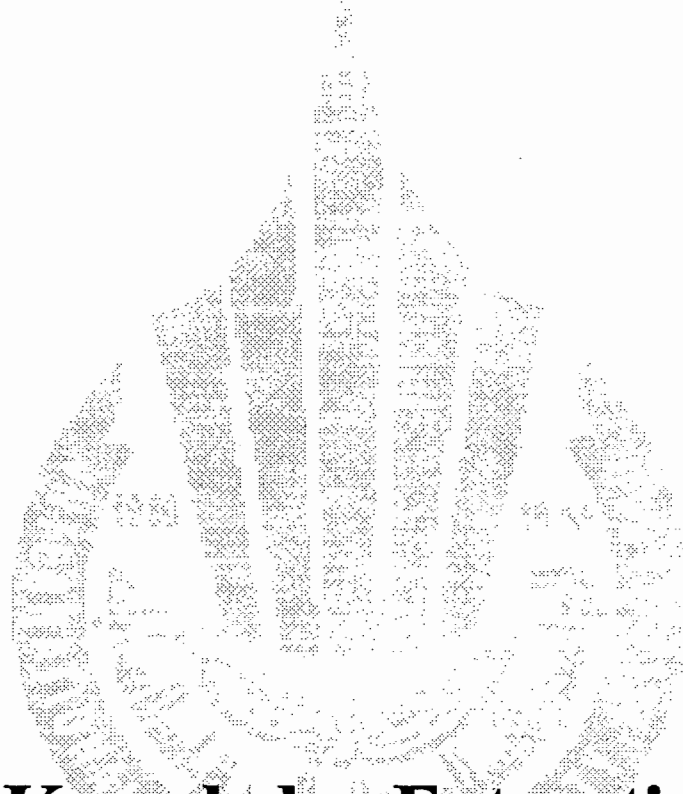


Figure 4.6: Decision Tree Transformed for knowledge Warehouse

CHAPTER 5



Knowledge Extraction Algorithm for Decision Tables

5.1 Introduction

A decision table is a simple way to organize knowledge for decision making. A decision table is organized in the form of a table consisting of rows and columns. The table has 2 parts. One is the list of all attributes that contribute in the decision making process and second is a list of all possible values for each attribute. Then a list of conclusions is developed. Finally different configurations of attributes are developed to match each conclusion [1].

5.2 Literature Review

Decision tables are used as a tool for organizing information for decision making. It can be viewed as a matrix in which upper rows specify sets of conditions and lower rows specifies the actions when some conditions are satisfied [36].

Moret in his article [36] said that each column of a decision table can be viewed as a rule which has the “if” condition and then the action to take when the condition is satisfied. Each column of the table represents a rule. If two overlapping rules specify different actions, they are called “inconsistent” and the table is called “ambiguous”. If they specify identical actions, they are called “redundant”.

Decision tables, like other quantitative models have three basic components. These are decision variables, uncontrollable variables and result variables.

Result Variables: These variables are the output of decision making. They represent the performance of decision making process. These are also called “dependent variables”.

Decision Variables: Are the alternative available actions. A knowledge worker calculates the values of all these variables for each alternative.

Uncontrollable variables: There are some variables which are not in control of the decision maker but they affect the decision variables. When values of these variables

are fixed, they are called parameters and when they can vary, they are called variables.

Relationship of these variables is shown in [1, pg46] as follows

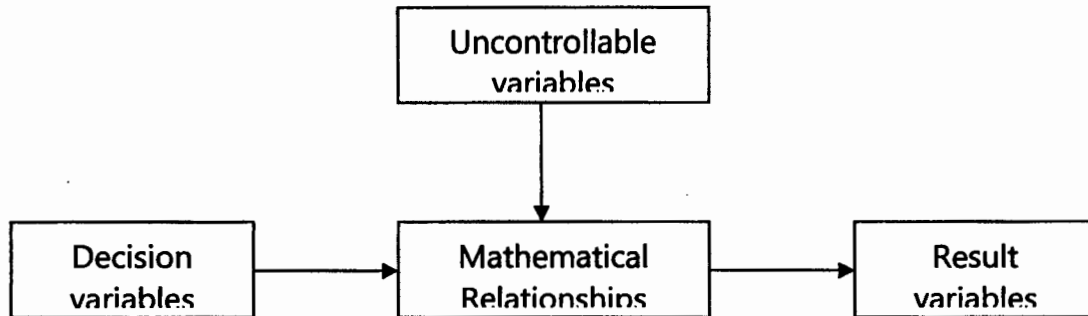


Figure 5.1: Relationships of variable [1, pg.46]

Turban has explored these types of variable giving their examples and area where they are used. It is formulated in the form of a table at page 46 of [1]. The table is reproduced as under

Area	Decision Variables	Result Variables	Uncontrollable variables and parameters
Financial Investment	Investment alternatives and amounts. How long to invest. When to invest	Total profit. Risk Rate of return Earnings per share Liquidity level	Inflation rate Prime rate Competition
Marketing	Advertising budget Where to advertise	Market share Customer satisfaction	Customer's income Competitors' action
Manufacturing	What and how much to produce Inventory levels Compensation programs	Total cost. Quality level. Employee satisfaction	Machine capacity technology. Material prices
Accounting	Use of computers. Audit schedule	Data processing cost. Error rate	Computer technology. Tax rates. Legal requirements
Transportation	Shipment schedule. Use of smart cards	Total transport cost. Payments float time. Customer satisfaction	Delivery distance. regulations
services	Staffing level	Customer satisfaction	Demand for services

Table 5.1: Types of Decision Variables [1, p40]

Use Examples of Decision Tables

Example 1: Here is an example of decision table from [5] to take the decision of how to spend a Saturday afternoon in a spring.

Raining?	Yes	No	No	Wind
Wind condition		Breezy	Calm	y
Clean basement	X			X
Spade garden			X	
Fly kite with children		X		

Condition Rows

Action Rows

Figure 5.2: a decision table on how to spend a Saturday afternoon in spring [5]

Top two rows in the table are condition rows one for each condition. First variable is a binary variable having values of “Yes” and “No” only while the second variable has three possible values of “Breezy”, “Calm” or “Windy”.

Bottom three rows are the action rows showing different available alternatives. As already discussed that according to Moret each column of the decision table can be viewed as a Rule. Hence this table has four rules. Labels of each action row are the available action which can be taken. In each column, if an action can be taken under the variable values specified in the column headers, the cell at the intersection of row and column is marked as “X”.

The four rules for the four columns of the table can be described as:

1. if it is raining then clean the basement
2. if it is not raining and wind conditions are breezy, then fly kite with children
3. if it is not raining and wind conditions are calm, then spade the garden
4. if it is windy then clean the basement

Example 2: Consider the example of an investment company given in [1] at page 179. The company uses a decision table for decision making and has three options for its investment. It can invest in bonds, stocks or certificates of deposits. The goal of decision making is to earn maximum profit. The list of attributes in this case is solid growth, stagnation and inflation. Values of these attributes are calculated by experts for each of the three alternatives.

The decision table given by Turban in [1] is given below

Table 5.2: A decision table for investment [1]

Alternatives	State of Nature (Uncontrollable Variables)		
	Solid Growth (%)	Stagnation (%)	Inflation (%)
Bonds	12.0	6.0	3.0
Stocks	15.0	3.0	-2.0
CDs	6.5	6.5	6.5

According to turban, this decision making process can be viewed as a two-person game. The investor makes the first move by selecting an alternative and then the state of nature makes the other move by the payoff as given in the decision table.

The decision table has decision variable, uncontrollable variables and result variables. Decision variables in this case are different alternatives; uncontrollable variables are stagnation and inflation. These are dependent on the state of economy.

This decision table can work very well if economy conditions are predictable and certain. But normally they are not, so we must consider two cases of uncertainty and risk as well. We do not know the probabilities of each state of nature for uncertainty but we can assume them for risk.

Algorithms for Converting Decision tables to Decision Trees:

The goal of this research work was to extract knowledge from decision tables and store them in a knowledge warehouse that will be in the form of a decision tree. For this purpose several algorithms for conversion of a decision table into a decision were studied. Following is a critical review of these algorithms.

I.K Sethi and B. Chatterjee Algorithm:

An algorithm for conversion of decision tables to efficient sequential testing procedures was proposed by I.K Sethi and B. Chatterjee in [32]. The scenario discussed is a sequential testing procedure for checking which rule from a decision table to apply in a particular scenario. According to [32] this can usually be found in the form of a minimum-path-length tree. So they developed an algorithm for converting decision tables into efficient decision trees. As the goal of the tree is to find the rule to apply with the minimum cost, the terminal nodes of the output tree will be the rules of input decision table.

Consider the input decision tree example from [32] in the figure below. To find the efficient algorithm, the probability of rules applicability is calculated as shown in top row of the table. Then testing cost against each condition is calculated as shown in the last column of the table.

Rule probabilities →	0.50	0.20	0.30	Testing cost
x_1	N	Y	—	4
x_2	N	Y	Y	8
x_3	N	Y	N	12
Actions →	A_1	A_2	A_3	

Figure 5.3: Rules Probability for decision table [32]

The output decision tree of algorithm given in [32] is shown in the figure below.

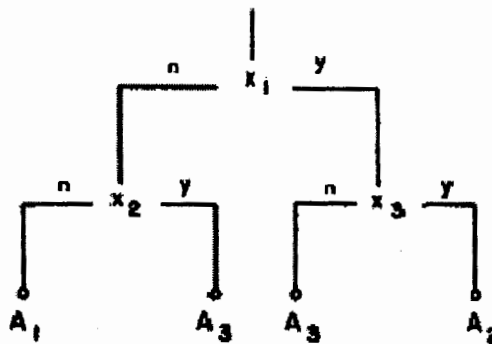


Figure 5.4: Decision table converted into a decision tree [32]

Notice that the actions are the terminal nodes of the tree and the conditions with most probability is evaluated first. The algorithm for making such decision tree is given below.

The algorithm has the following assumptions:

1. Conditions of the decision table can be tested in any sequence
2. Each condition testing result in either Y or N.
3. The cost for testing each condition is specified.
4. rule probability is either known or can be calculated
5. the decision table is not ambiguous

The algorithm is given as under

Step 1: select the first condition of the decision table as a possible candidate for the root node and construct two sub-tables for each Y and N conditions.

Step 2: try to merge the rules which result in the same action set in the two sub-tables.

Step 3: calculate the estimated cost for each of the two subtables

Step 4: calculate the estimated cost of the complete tree keeping the chosen node condition as root node

Step 5: repeat the same steps for all conditions to select a condition i with the minimum cost

Step 6: keep the condition x_i at the root node and make the Y and N subtables. Repeat above steps on each subtables until the tree is complete.

S. Ganpathy and V. Rajaraman Algorithm:

This algorithm was developed for coding a decision table into a computer program. A decision tree is a good option to code as a program so the program converts the decision table into a decision tree. The algorithm is also suited for converting decision tables to flowcharts.

Just like the previous algorithm, each node of a tree is a condition having two possible paths for Y and N. all the terminal nodes are the actions to perform while all condition nodes test a condition based on the tree logic.

Some definitions are given and explained with the help of following picture

	<i>R1</i>	<i>R2</i>	<i>R3</i>	<i>R4</i>	<i>R5</i>
<i>C1</i>	<i>Y</i>	—	<i>Y</i>	<i>N</i>	<i>N</i>
<i>C2</i>	<i>Y</i>	<i>Y</i>	<i>N</i>	—	<i>N</i>
<i>C3</i>	—	<i>N</i>	—	<i>Y</i>	<i>N</i>
<i>A1</i>	<i>X</i>	<i>X</i>	—	—	—
<i>A2</i>	—	—	<i>X</i>	—	—
<i>A3</i>	—	—	—	<i>X</i>	<i>X</i>

Figure 5.5: Decision table format [19]

In the table above, C1, C2 and C3 are conditions, A1, A2 and A3 are possible actions whereas R1, R2, R3, R4 and R5 are the rules.

Elementary Rules: Each element of a decision table can have one of Y, N or a – (dash) entry. A rule which has p dash entries can be expanded into 2^p rules without – entry. A rules having only Y and N entries and no – entry is called an elementary rule. A decision table with n conditions will have 2^n elementary rules.

Action Set: All the rules which lead to identical actions make action set

Overlapping Rules: Two rules are called overlapping if they have one or more common elementary rules.

Disjoint Rules: If two rules do not overlap then they are called disjoint

Ambiguous Rules: If two rules from different action sets overlap then the decision table is ambiguous.

Redundant Condition: The elementary rules within an action set may be combined using Boolean algebra. However there is no unique way to combine elementary rules to arrive the set of rules. If elementary rules in a decision table can be combined to yield all dash entries for a condition row then that condition is redundant.

Totally relevant condition: if elementary rules in an action set cannot be combined to yield a dash entry for a condition, then that condition is relevant for that action set. In the same research [19] some statements for conversion of decision table to trees are made. These conditions are given below

-
1. If a set of conditions is totally relevant, then any of these conditions could be tested first. The sequencing among a set of totally relevant conditions does not affect the optimality of the tree.
 2. A redundant condition should not be tested.
 3. The above statements would also apply to any subtable obtained during the development of an optimal tree.
 4. If there exists a rule(s) in an action set such that it does not combine with any other rule in that set to yield a dash entry for any condition, then that rule could be considered to constitute a separate action set(s) without affecting the optimality of the resulting tree. In other words the same tree will be optimal for both tables.
 5. Yes and No entries should be treated symmetrically.
 6. The Else Rule should be treated just like any other action set.

Assumption for the Ganapathy and Rajaraman Algorithm:

Following assumptions were made by Ganapathy and Rajaraman for their algorithm

1. The condition in a decision table can be tested in any order
2. The outcome of every condition is either Y, N or – (dash)
3. The time for testing each condition is known
4. The probability of occurrence of each elementary rule is known or can be calculated
5. The decision table is not ambiguous

Ganapathy and Rajaraman Algorithm:

The algorithm proposed by Ganapathy and Rajaraman in [19] for conversion of decision table into a decision tree is given below

Step 1: Expand all the rules into elementary rules

Step 2: Use combining techniques on elementary rules for all the conditions separately to maximize the number of – entries for each condition. This step will give a decision table for each condition

Step 3: For each decision table, calculate sum of probabilities of Y p_1 and sum of probabilities of N p_0 for all the conditions. If there is a condition where $p_0 + p_1 = 1$ then go to step 7, else go to next step.

Step 4: Calculate the following values for each condition in the tables obtained in step 2.

$$p_Y = p_1 / (p_1 + p_0);$$

$$p_N = p_0 / (p_1 + p_0);$$

$$H(k) = (-p_Y \log_2 p_Y + p_N \log_2 p_N),$$

$$I(k) = (p_1 + p_0)H(k),$$

$$L(k) = H(k) - I(k), \text{ and}$$

$$I(k)/T(k).$$

Find the condition which gives maximum value of $[I(k)/T(k)]$ as the condition to be tested. If more than one conditions give the same value of $[I(k)/T(k)]$, then pick the condition with minimum $L(k)$. if all have same value of $L(k)$, then pick condition with minimum $T(k)$.

Step 5: use the selected condition as the current node and split the decision table into the Y and N halves.

Step 6: there is no need to test any further if all the entries are – entries. If there is one condition in the subtable the test that condition and if there are more than one conditions in the subtable, go to step 2.

Step 7: pick all condition with $p_0 + p_1 = 1$ and make the tree.

5.3 Problem Scenario for Decision Tables ETL Algorithm

As discussed in the previous chapters, knowledge needs to be extracted and transformed from knowledge documents before storing it in a knowledge warehouse. Decision tables have widespread applications and have been used to document organization knowledge in an easily manageable form.

There is a need to devise an algorithm to extract knowledge from decision tables and transform it in a format compatible with knowledge warehouse storage structure. For this research the storage structure of the knowledge warehouse is object oriented decision trees. The structure of the object and other details about storage has been discussed in the previous chapters.

An algorithm is proposed below for transformation of knowledge extracted from decision table into the proposed knowledge warehouse structure. This algorithm will take input a decision table and make necessary transformations to store the knowledge in a knowledge warehouse in accordance with the storage structure proposed earlier in this thesis.

5.4 The Proposed Algorithm for knowledge ETL from Decision Tables

Assumptions:

Following assumptions are made for this algorithm:

1. The conditions in the decision table can be tested in any order
2. The result of testing a condition is either Y, N or –
3. The decision table is not ambiguous

Proposed Algorithm

// Comment: make rules

For each column in the decision table

For each condition in the Decision Table

If rule needs this condition satisfied

Mark entry as "Y"

If rule needs this condition Not satisfied

Mark entry as "N"

If this condition has no effect on rule

Mark entry as "-"

Next condition

Next column

// Comment: Select Condition for Root Node

//Comment: if one condition is found at any Node of the forest, select that condition
for root node

If more than one such nodes exist, select the node at highest level

Variable SelectedRootNodeID = NULL

For each condition in decision table

If condition exists at any root KWNode of the forest

If SelectedRootNodeID is NULL OR

KWNode. Level > SelectedRootNodeID.Level

SelectedRootNodeID = KWNode

End if

End if

Next condition

**//Comment: If a node is not found in the warehouse, select a condition for root node
and create node for it**

If SelectedRootNodeID is NULL

Select the condition with maximum – entries for root node

Write code for testing condition in decision function

Set node variables

SelectedRootNodeID = newly created node

End if

//Comment: create rest of the tree

For all rules in the decision table sorted by maximum - entries

For all conditions in the rule

Parse the tree according to affirmative and negative child nodes

If there is no node for this condition

Create the node and set its parent accordingly

Write code for decision function

End if

Next condition

Next rule

5.5 Case Study for Decision Tables ETL Algorithm

Consider the decision table given below as a case study for this algorithm.

Raining?	Yes	No	No	Wind
Wind condition		Breezy	Calm	y
Clean basement	X			X
Spade garden			X	
Fly kite with children		X		

Figure 5.6 : decision table for case study [1]

The first step of the algorithm makes rules from the given decision table. It will mark the appropriate entry with “Y”, “N” or “-“ depending on the contribution of that condition in a specific rule. A “Y” entry will depict that this condition must be specified to apply the rule. A “N” entry will depict that this specific condition must not be satisfied to apply this rule and a “-“ entry will depict that this condition has no effect on the rule. When applied to the decision table given in figure 6.6, the rules made are shown in form of a table below

Table 5.3: Rules Made from the given decision table

Condition	Rule1 (Clean Basement)	Rule2 (Spade Garden)	Rule3 (Fly kite with Children)	Rule4 (Clean Basement)
Raining	Y	N	N	-
Breezy Wind	-	N	Y	N
Calm Wind	-	Y	N	N
Windy	-	N	N	Y

In the next step of the algorithm, Rule1 will be selected for root node as it contains maximum “-“ entries. After that the nodes with maximum number of “-“ entries will be selected for next nodes of the decision tree. Actions will become leaf nodes of the tree and decision functions for these leave nodes will perform the action specified.

The final outcome which is in the proposed storage structure after applying step 3 of the proposed ETL algorithm is given in figure below.

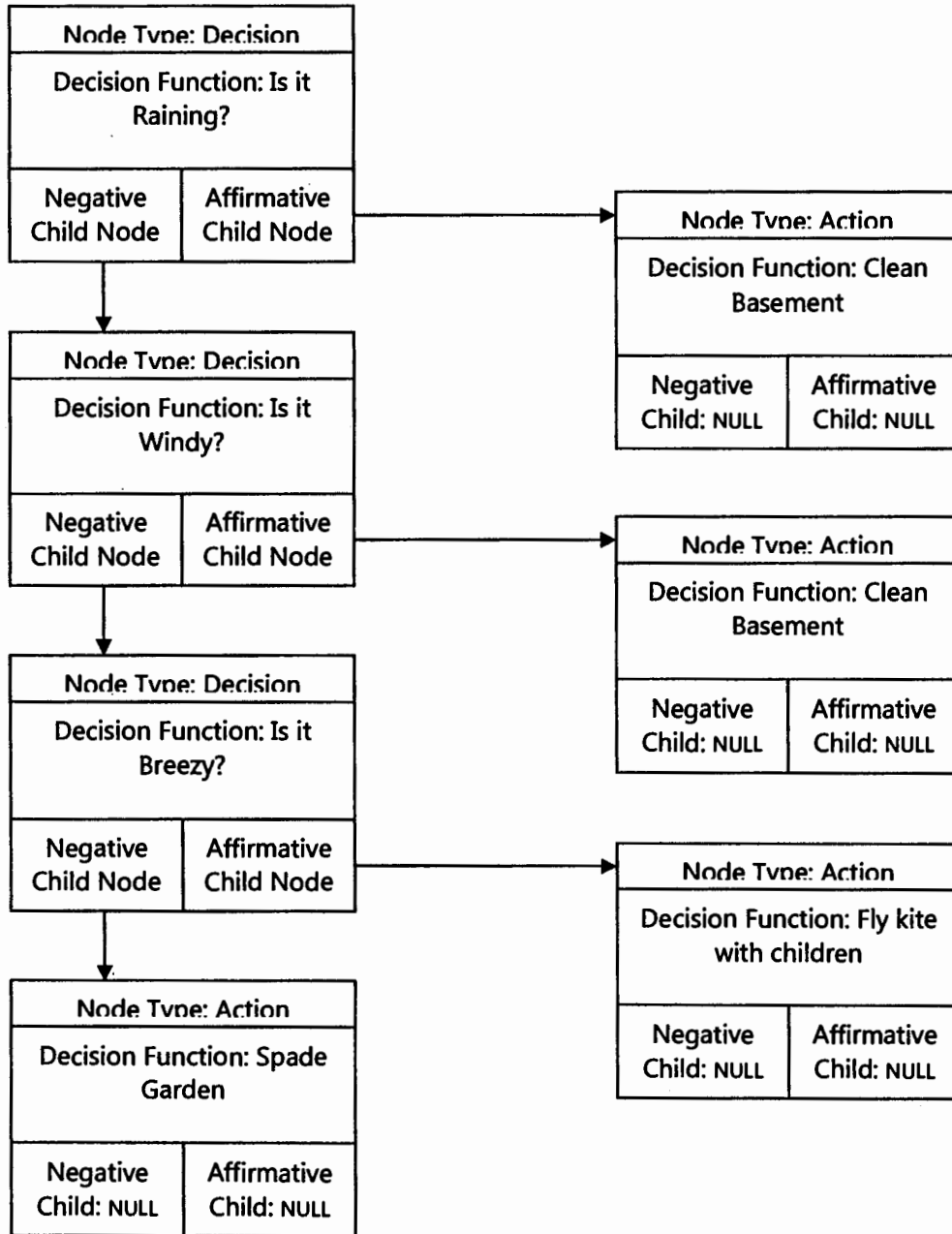


Figure 5.7: Decision table transformed for knowledge warehouse

5.6 Implementation of Algorithm

Simulation results of the case study presented in preceding section is shown in the figure below

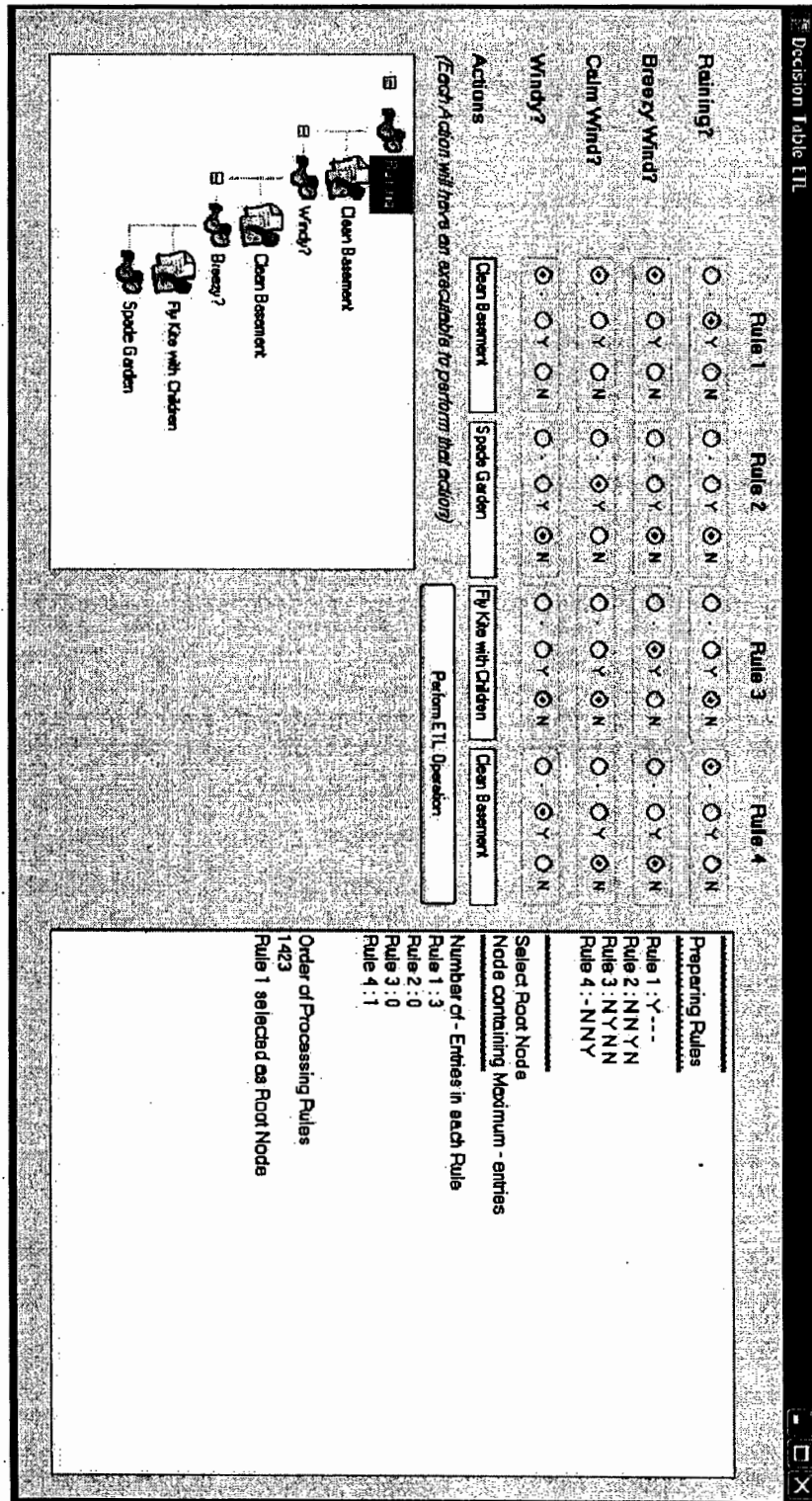


Figure 5.8: Simulation result of Decision table ETL Algorithm

CHAPTER 6



Knowledge Extraction Algorithm for Use Cases

6.1 Introduction

Use cases are used to document the scenarios of system usage. They are developed in the system requirement analysis phase of software engineering to describe the system usage.

Use cases describe how the system will behave in response to an event from outside the system. They describe the system from the user's point of view.

6.2 Overview of Use Case Documents

Overview:

Use cases describe the interaction between an actor and the system in a sequence of steps. Use case development starts with identification of Actors. Any people or devices that play a role in the system are called Actors. It can be any external object which interacts with the system. Mostly an actor is considered a synonym of user where as a user can play number of roles when interacting with the system. Other devices which interact with the system may also be an actor.

UML standard for use case diagrams are used to represent a use case in a standard graphical representation. Use cases are normally written in plain language so that non-technical users can give an input in the development phase.

Following are the suggested questions that should be answered by a use case diagram [22].

- What are the main tasks or functions that are performed by the actor?
- What system information will the actor acquire, produce or change?
- Will the actor have to inform the system about the changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

History of Use Cases:

The first modeling technique for use cases was formulated by Ivar Jacobson, in 1986[23]. He later contributed on UML as well. Later use case modeling was improved by Kurt Bittner, Alistair Cockburn, Gunner Overgaard and Geri Scheneider.

Use cases were popular in requirement gathering for software engineering by the 1990s. They were mostly popular in object oriented systems but were also very useful in other non-object oriented systems.

Types of Use Cases:

Business Use Cases: Are used to describe a process for providing a service to a business actor. It is normally written in non-technical language [23]. Examples of business use cases include manual payment processing, manage corporate real estate.

System Use Cases: Are used to describe system functionality to specify to functions and services given to the user by the system. For example create voucher is a system use case [23].

Use case Format:

There is no standard template for documenting use cases. Some times use cases include more or less details depending on the type of application and some times they may contain industry specific sections [23]. Sections of a typical use case and their brief descriptions are given below.

Use case name: This is a unique identifier given to each use case. It is normally in the verb-noun form for example “Borrow Books”, “Withdraw Cash” etc [23]. Normally it is chosen in a way that it describes completely what the use case is about.

Version: Use cases mature during the different phases of software development life cycle. A version tells the reader at which stage of SDLC, this use case was developed. Older versions are also kept and used for reference.

Goal: This section tells the actual goal which is intended to reach by the user in this use case. Every use case must have a goal which is the final outcome the use case is intended to provide.

Summary: This section contains a few line summary of what the use case is about and what it intends to achieve. It facilitates the reader to get the essence of use case without reading complete use case.

Actors: This section documents the actors participating in the use case.

Preconditions: This section has the conditions which must be met before the use case is triggered. The system needs to be in a specific state to initiate a use case. This state is described in the preconditions section.

Triggers: This section describes the events that initiate the use case. This event can be internal, external or temporal [23].

Basic Course of Events: This is the main section of a use case and is documented as a set of steps taken by the system or actor. It is also called “basic flow”, “happy flow” or “happy path”.

Alternative paths: There may be some variations in the normal path of the basic course of events. These are termed as secondary paths or alternative scenarios. Whenever a test condition arrives in the use case events, there will be an alternative path to handle that scenario.

Post conditions: This section contains the state of the system after the use case is complete. These conditions are guaranteed to be true at the end of use case.

Business Rules: This section describes the normal working scenarios of the organization specific to the use case.

Notes: This section contains any information author want to document and it does not fall into any other section.

Author and Date: It is necessary to document the author of the use case so that he can be consulted during any phase of use case implementation. Dates are also very important as there are normally many versions of a use case during its life cycle.

6.3 Literature Review for knowledge extraction algorithm from a use case:

The goal of this research is to extract knowledge from use cases and transform and load them into a form that can be processed by machine. During the literature review emphasis was on looking for existing algorithms to extract data from use cases and store them in knowledge form. The two most relevant research works are discussed below.

Converting Use Cases to System Sequence Diagrams:

An algorithm is proposed in [20] for using use-case models for reliability assessment. Reliability of a system is defined as the probability for a failure free operation of the system for a specified time with a set of operating conditions[20]. Manual approaches for reliability assessment were completely dependent of the analysts and knowledge workers and hence were not accurate. They had their drawbacks specially when dealing with unfamiliar systems or very large complex systems.

The proposed algorithm discussed in [20] has three basic steps

1. Convert use case into an SSD (System Sequence Diagram)
2. Convert SSD into a UCG (Use Case Graph)

3. Determine the reliability metric

This research is about extracting knowledge from use case documents so only the step 1 of the research is relevant to this research.

The algorithm given in [20] is given below

- a. For each step specifying a user input UI , add a message MU from user to system in SSD and name the message MU same as the input UI .
- b. For each step describing a system response SR, add a message MS in SSD from system to user, and name the message MS same as the system response SR.
- c. For all steps describing system action, no message is included in SSD.
- d. Preserve sequence of user input UI and system response SR in SSD same as in the scenario description.

This algorithm exploits the fact that use cases and sequence diagram show interaction between the system and user. Steps performed by the system and actor to complete a use case scenario are listed in a scenario in use case models and the same is done in a sequence diagram where actions performed by the system and external interactions with the system.

Converting use cases to state diagrams

A technique is presented in [21] to automate the synthesis of state diagram from a use case. This approach takes usecase as input and converts it into a sequence diagram. As an intermediate step the usecase is converted into a tree representation. The tree representation in this case is not a decision tree and discussed the tree as a sequence of messages passed between actors of a system which makes the modes as states for further conversion into a state diagram. Here the portion of representing a use case in tree form will discussed as it is the part relevant to this research.

A use case is a set of possible scenarios. It is hard to automate such form of use case hence it is converted into a tree representation. The tree representation of a “Make a call” use case as discussed in [21] is given in the figure below.

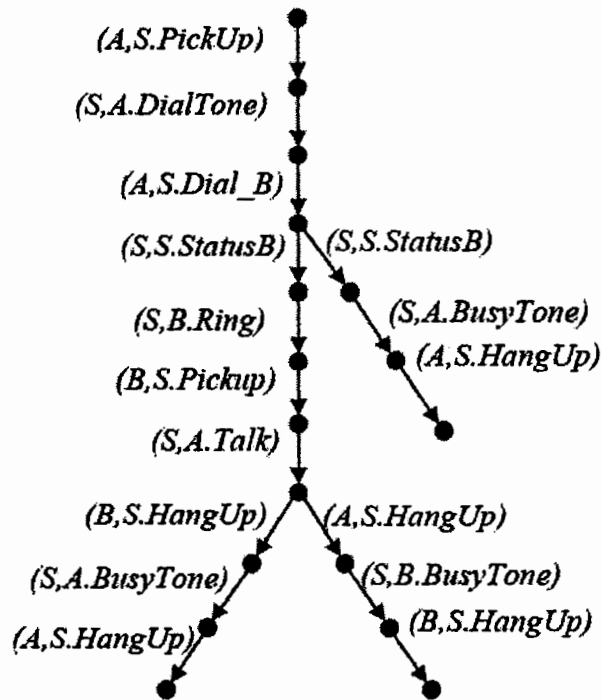


Figure 6.1: Make a call use case represented as a tree [21]

This figure shows tree representation of use case “Make a call” for a basic telephone system. Each node of this tree represents a message in the form (O_i, O_j, m) where message m is sent from object O_i to O_j .

Using this technique a use case can be represented in a tree form in which each possible scenario is a path from root to leaf. Author has discussed in [21] that a sequence diagram can be made for each of these paths.

6.4 Problem Scenario for knowledge ETL from Use Cases:

Output of this research is to devise an algorithm to extract knowledge from usecase documents and store it in the form of a decision tree so that it can be used for automated decision making.

Different algorithms were studied to store the usecase in a form which is computer readable but none of them were suitable in a knowledge warehouse perspective. We have already devised a structure for a knowledge warehouse node which can

accommodate executable code and can also store other information required or decision making. The algorithm designed to load a use case document in this knowledge warehouse decision tree structure is given below.

6.5 Proposed Algorithm for Knowledge ETL from Use Cases

Store information like system, use-case ID, usecase name, actors, priority, stakeholders, goals, relationships, input, output, test cases etc in meta knowledge

Create a root node

Write function of this root node to test precondition

Create a node for failed state

Write code in failed state function to return the status

Set failed node as negative child of root node

For its affirmative child create the rest of tree

For all steps in the scenarios grouped by steps performed by same actor

 If group of steps is performed by actor then

 Create node for this step and write the code to take input from actor as given in the steps. Set the negative child node to failed state

 Else if group of steps is performed by system then

 Create node and write function to perform the task. Set the negative child node to Failed State

 End if

 If an alternate and exception flow is available for this step

 Create a negative node to accommodate this alternate flow

 For all substeps in alternate flows

 Create node and write function for substep

```

    If action = "RESUME" then
        Set child node to the next affirmative node
    Else if action = "FAIL" then
        Set child node to fail state
    End if
End for sub steps
End for all steps

If post conditions exist then
Create a node for post conditions as affirmative child and write function. Set
affirmative child node to accept state
End if

```

6.6 Case Study for Proposed Use Case ETL Algorithms

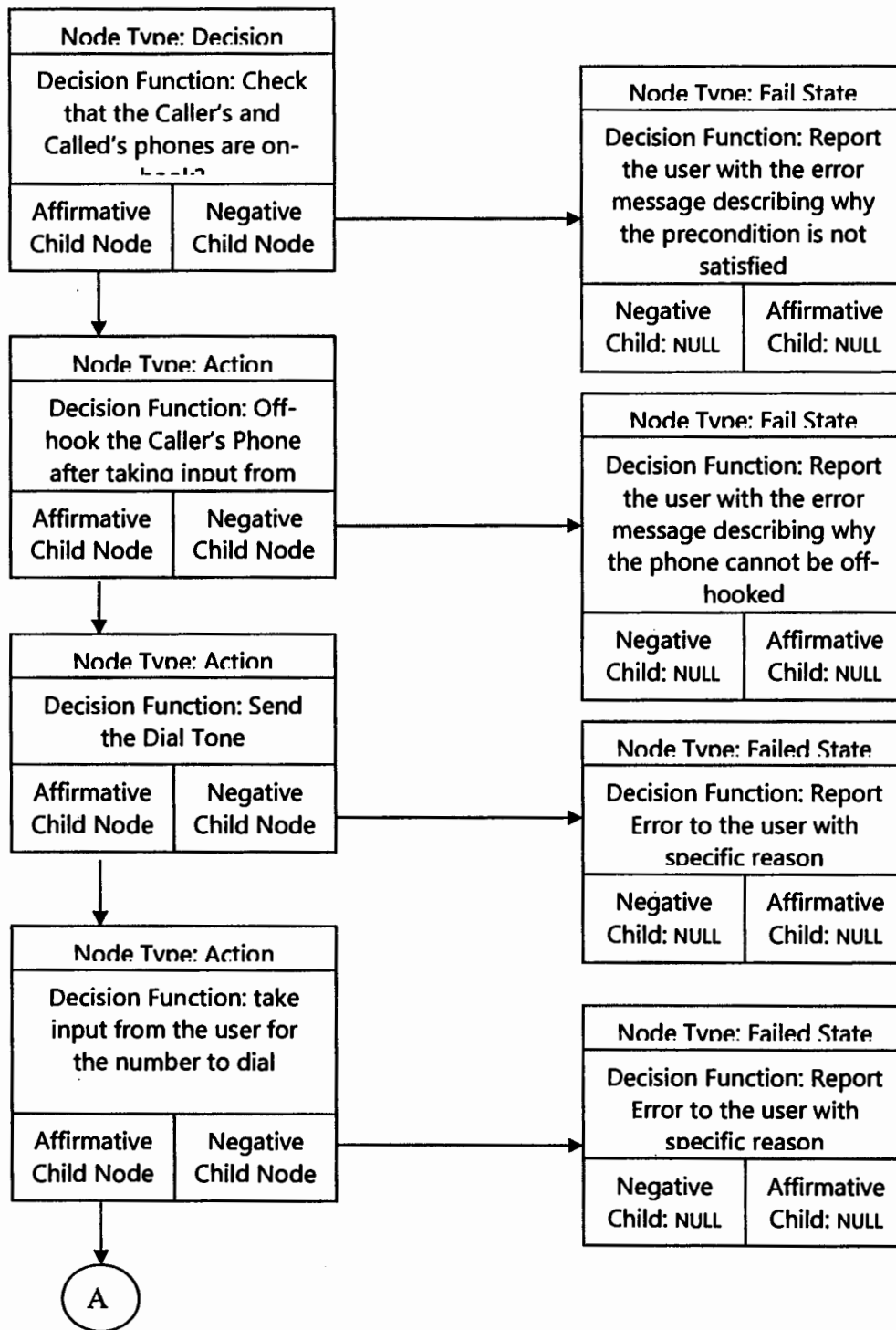
Consider the following use case for making a telephone call. In this section it will be treated as a case study for algorithm validation purpose

<p>Make a Phone Call Primary Actors: Caller--Person making the call Called--Person receiving the call Supporting Actors: Switching Network--Equipment handling calls between telephones Level: User Goal Preconditions: Caller's and Called's phones are on-hook. Success Guarantee: Switching Network has broken down call, and all components are in the same state they were before call. Main Course:</p> <ol style="list-style-type: none"> 1. The use case begins when Caller takes phone "off-hook." 2. Switching Network sends dial tone. 3. Caller dials call. 4. Switching Network translates digits. 5. Switching Network connects call to Called's phone. 6. Switching Network sends ring signal to Called's phone and to Caller's receiver (so that they can hear it ringing). 7. Called answers phone and talks to Caller. 8. Called and Caller hang up. 9. The Switching Network breaks down the connection, Logs Billing Information, and the use case terminates.

Figure 6.2: Use case for making a Phone call [34]

This is a simple use case for making a telephone call. The 2 persons on a telephone conversation are the primary actors and the switching network is working as a supporting actor.

The subject usecase when loaded in the knowledge warehouse according to the proposed algorithm can be graphically represented as follows.



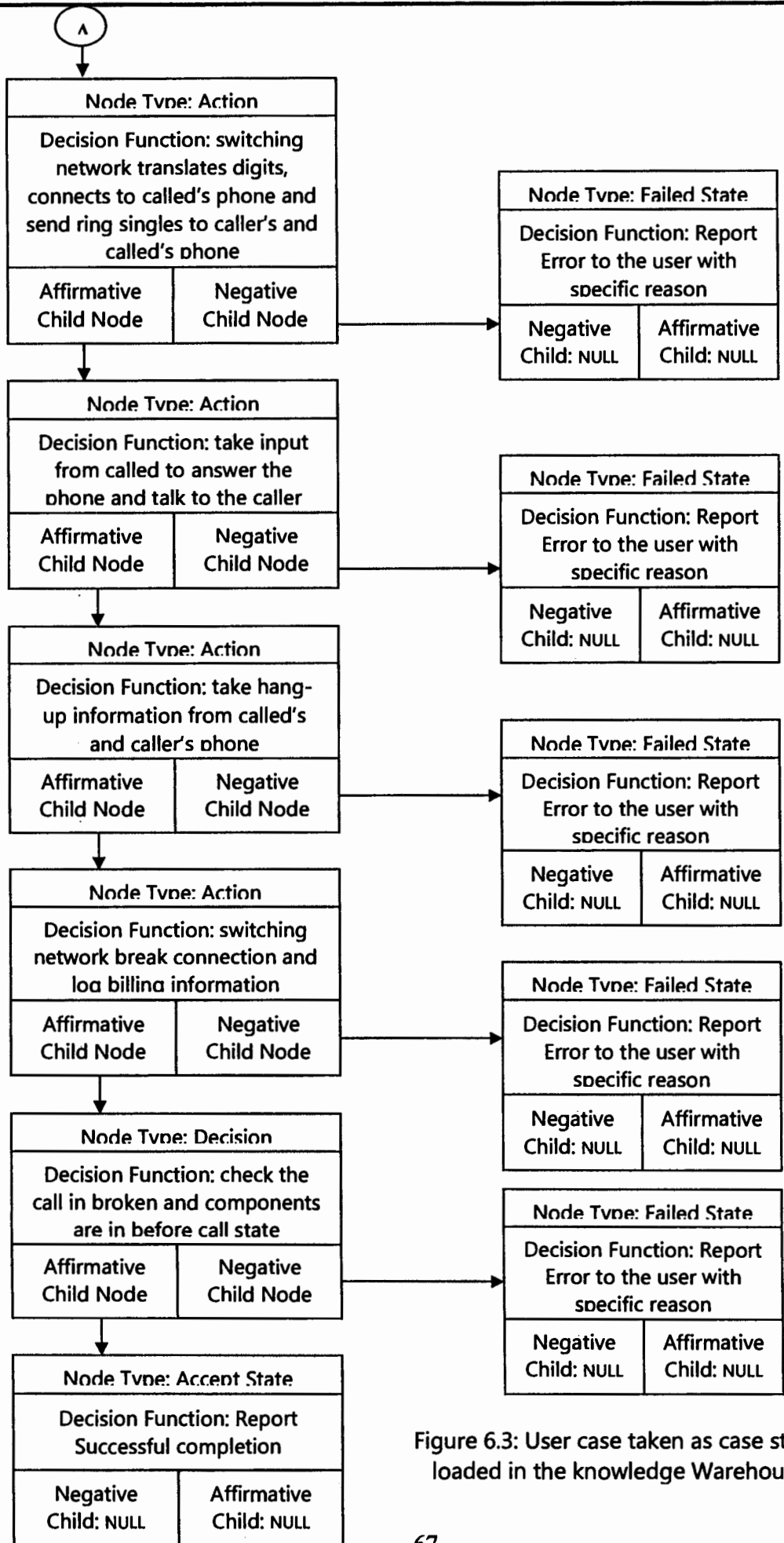


Figure 6.3: User case taken as case study loaded in the knowledge Warehouse

6.7 Implementation of Algorithm

Simulation results of the case study presented in preceding section is shown in the figure below

The screenshot displays the JSpecs IT Simulator interface for the use case 'Make a Phone Call'. The main window is divided into several sections:

- User Case Name:** Make a Phone Call
- User Goal:** (Empty)
- Primary Actors:**

Actor	Description
Caller	Person making the call
Called	Person receiving the call
- Supporting Actors:**

Actor	Description
Switching Network	Equipment routing calls between telephones
- Precondition:** Caller's end Caller's Phone is on-hook
- Event Table:**

Event	Actor
Off-hook the phone	Caller
Send dial tone	Switching Network
Dial tone	Called
Transfer digits	Switching network
connect to called's phone	switching network
send ringing signal to caller and called...	switching network
answer phone and talk	Caller
- Processing Log:** Shows system messages such as 'Node state for event id 1 with its FILL state node Actor changed: Done this node'.
- Output Tree:** A hierarchical tree of messages and state changes, including 'M check caller and called ring tone', 'M off-hook caller phone', 'M send dial tone', 'M got number to dial', 'M forward called one send ring tone', 'M answer phone call', 'M later transfer information from caller and', 'M bank connection and logging', and 'M request answered'.
- Positional Data:** A section for defining the initial state of the system, currently showing 'NO pool condition'.

Figure 6.4: Simulation result of Use case ETL Algorithm

CHAPTER 7



Knowledge Extraction Algorithm for Knowledge Maps

7.1 Introduction

Knowledge maps are important knowledge documents and organization use it for knowledge management purposes. It is a subdivision of knowledge management and falls in the category of mapping sciences[29]. Use of computer sciences has made this knowledge management tool more powerful. It is also very good for knowledge generation as many authors agree that more and more relations are discovered in the knowledge is formulated in the form of a map[29].

Goal of this work is to extract knowledge from a knowledge map for loading into a knowledge warehouse. The knowledge warehouse storage structure for this research is in the form of an object oriented decision tree. The knowledge map technique will be discussed in the perspective of knowledge stored in the map and on converting it to a decision tree.

7.2 Literature Review for Knowledge Maps:

Definition:

Knowledge map is defined as visual representation that established a landscape, or domain, names the most important entities that exist within that domain and simultaneously places them within two or more relationships.[29].

Knowledge maps in defined in [26] as a union of three object. $K = U \{S, G, T\}$ where

1. S stands for support and represents a domain of plane or space. This plane can be in the form of any geometric shape. It can be in 2-D or 3-D. its boundaries may be marked or unmarked.
2. Graphics G is a part of mathematical theory of graphs which deals with the properties of shape and their geometrical formations.
3. T stands for Text and represents the text chains or alphanumeric characters attached to graphics G.

Where S is for Support and describes the domain of plane or space, G is for Graphics and represents the set of plane r geometrical objects and T is for Text and describes a set of alphanumeric signs.

Advantages of knowledge map

Knowledge map is an effective knowledge management tool due to the following reasons [31]

1. It is not a knowledge set but a guide. It is not the knowledge but tells the origin of knowledge.
2. As it only stores the information about source of knowledge so it is a powerful tool for tacit knowledge in some person's memory.
3. It can not only point to the sources of tacit knowledge but also relations between knowledge sources.

Knowledge mapping includes identifying, collecting, reviewing, validating, storing and sharing knowledge and information. Just like a geographical map it guides user to help him locate required knowledge which can be in tacit or explicit form [29].

Knowledge map is mainly provides the assistance to easily and effectively locate knowledge.

Types of Knowledge maps

1. Hierarchical or Radial Structure Maps, Concept Maps and Mind Maps

These maps organize knowledge in network or hierarchies of concepts just like the functioning of human brain. They provide a model for hierarchical structure from top level to bottom. The mapping process starts with a central concept in the middle and grows towards more specific topics.[29]

2. Casual maps

This type of knowledge map is also known as concept map, mind maps or manual knowledge maps.[28] This is a technique in which strategic thinking and acting are connected to make sense of complex problems [29]. These maps are in the form of graphs in which nodes represent the concepts that are linked through edges.

These maps try to codify the cause and effect knowledge[29] which is very helpful in business decisions just like something happens after some other thing has happened.[29]

An example of casual map of environmental forces and characteristics of technology taken from [29] is given below

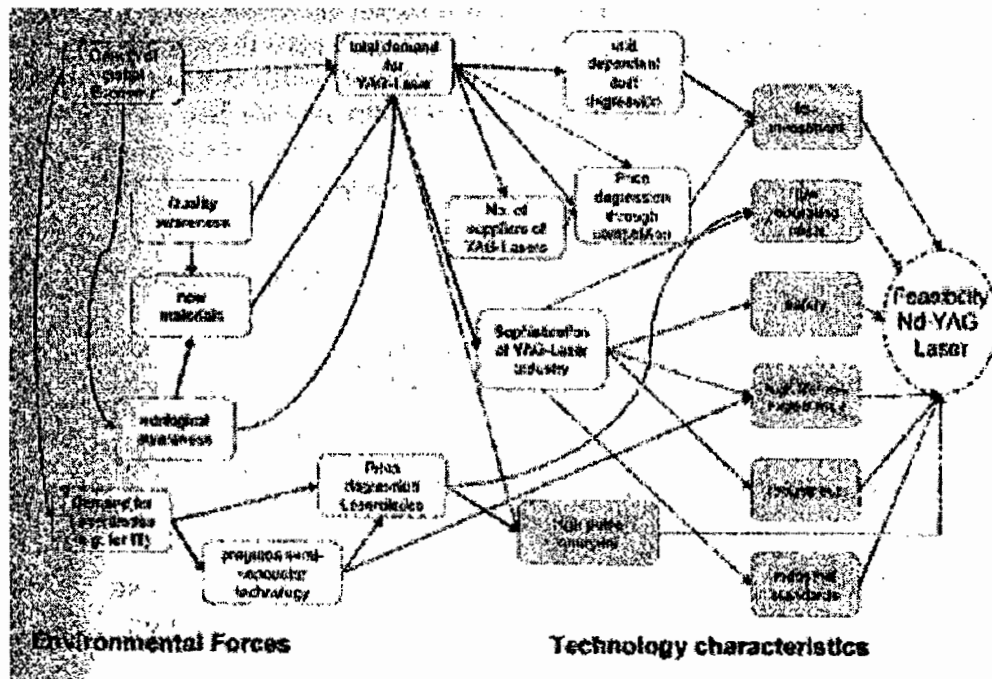


Figure 7.1: causal map of environmental forces and characteristics of a technology [29]

As a use case example taken from [28], consider a research conducted for automatic generation of a hierarchical knowledge map based on online Chinese news specifically for finance and health news. This map can highlight the current major news topics and relationships between them by showing concepts as blocks and connecting these blocks with lines to show a relationship between concepts. .

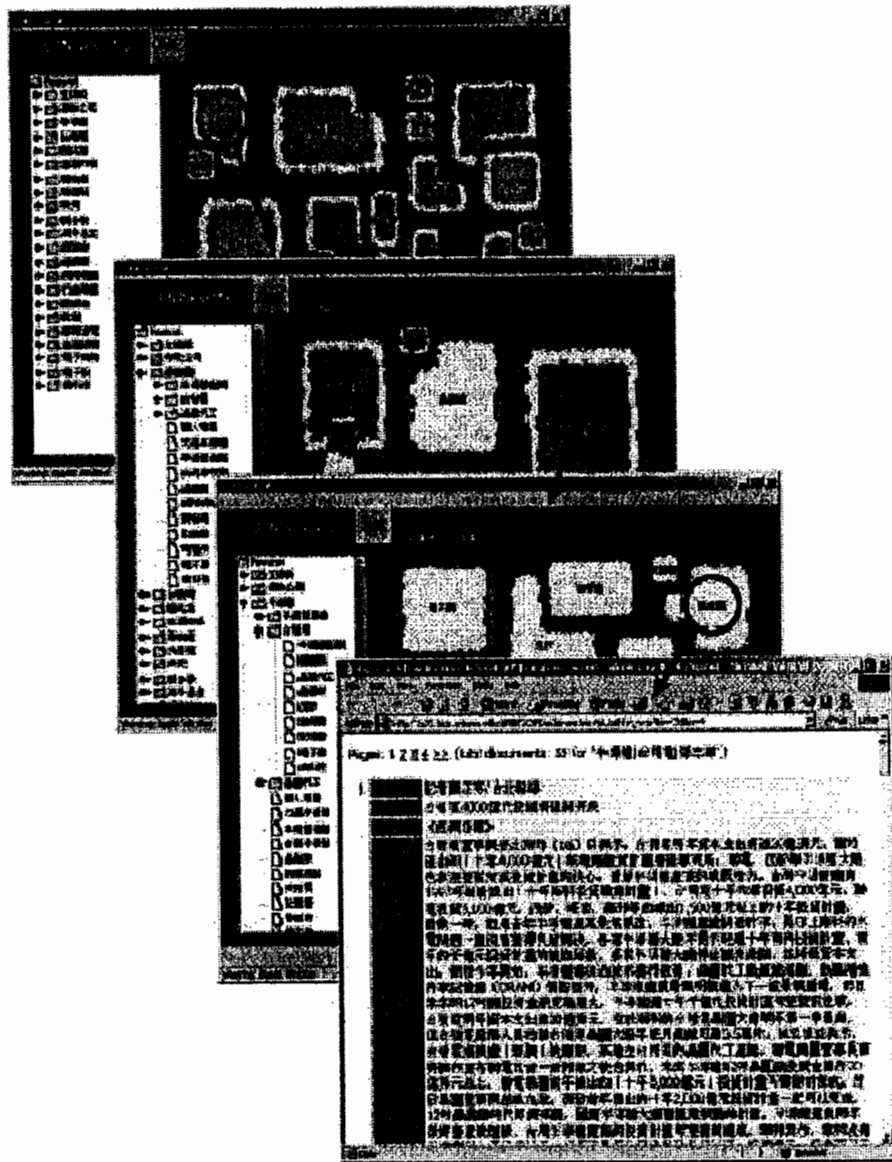


Figure 7.2: knowledge map visualization for a news map [28]

3. Knowledge source maps

Knowledge source maps focus on expertise in the organization. They guide to locate experts of a specific domain. An example of knowledge source map is given below.

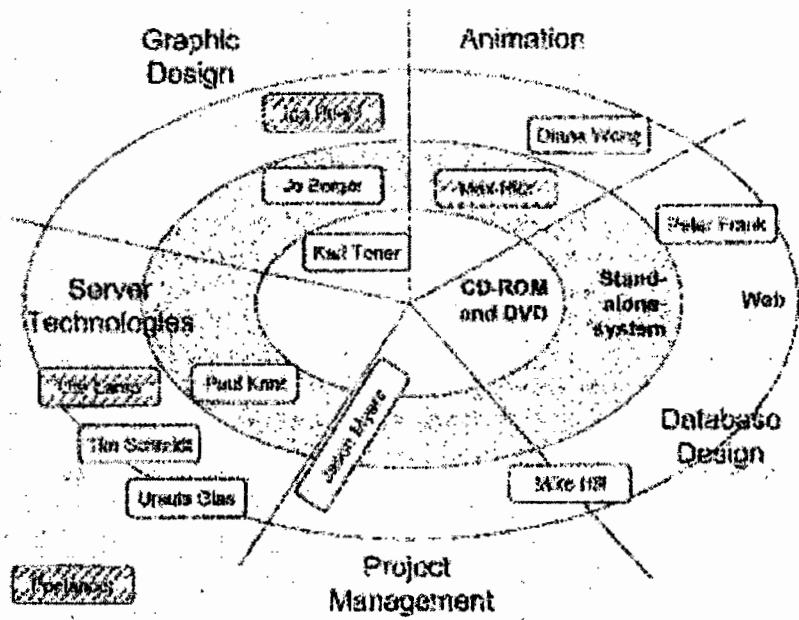


Figure 7.3: Knowledge source map [29]

4. Knowledge flow maps

These maps show the step by step process of doing something in phases. An example is given in the figure below [29]

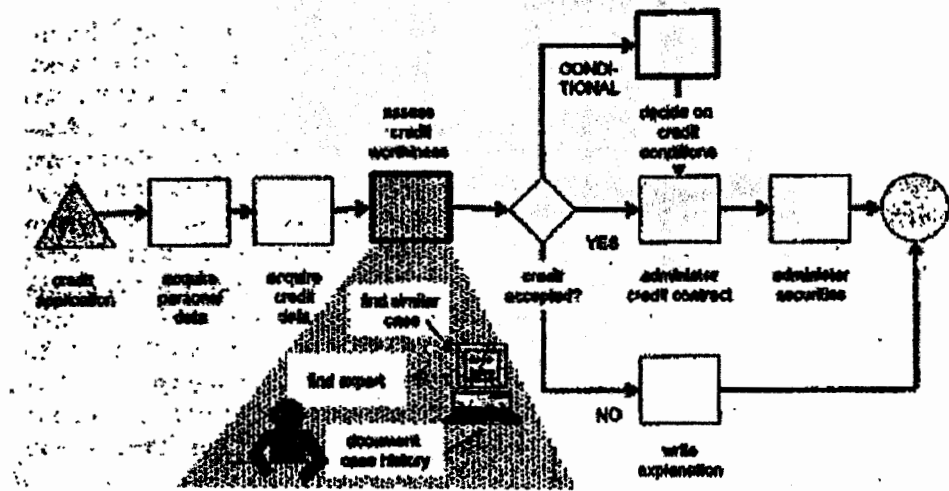


Figure A.3 Process flow and knowledge flow map

Figure 7.4: process flow and knowledge flow map [29]

5. Subject Hierarchy Maps

This knowledge map contains an alphabetical list of topics organized in groups and subgroups to form a hierarchy. This map misses a graphical representation but it is an effective way for classification of information. Telephone directories and yellow pages are examples of this type of knowledge map [28].

These are just some examples of knowledge maps but as we go deep in knowledge maps and study knowledge maps developed for different uses it reveals that knowledge maps can be of many other forms. Lets discuss some use cases to discover more about knowledge maps implementations.

7.3 Problem Scenario for Knowledge ETL Algorithm from Knowledge Maps

The goal of this research is to device an algorithm to extract knowledge from a knowledge map and load it in a knowledge warehouse. As discussed earlier knowledge maps can be of many different types. All have different data and represent knowledge differently.

The knowledge map format followed for this research is given by Chain Zins in [24]. He proposed a knowledge map for information sciences in which all the fields of information sciences were categorized with examples in a knowledge map. The map presented by him is given in the figure below.

Domain	Focus	Main Categories (1 st division)	Sub-Categories (2 nd division)	Sub-Categories/Examples & Explanations** (3 rd division)	Exemplary Fields	
Meta-Knowledge	Knowledge on the field/object	1. Foundations		Theory	A. Conceptions B. Disciplines (e.g., Anthropology (e.g., "culture"), Arts (e.g., "design"), Communication (e.g., "communication", "media", "message"), Computer science (e.g., "computer language"), Economics (e.g., "transformation economics"), Education (e.g., "learning"), Engineering (e.g., "information technology"), History (e.g., "primary source", "secondary source", "tertiary source"), Law (e.g., "intellectual property", "copyright"), Linguistics (e.g., "language"), Philosophy (Epistemology (e.g., "knowledge"), Ethics (e.g., "information ethics", "professional ethics"), Political Science (e.g., "democracy"), Psychology (e.g., "cognition"), Research Methodology (e.g., "evaluations", "research", "research methodology"), Semiotics (e.g., "sign"), Sociology (e.g., "society") C. Theories	Theory of IS
				Research	A. Theoretical B. Empirical 1. Quantitative 2. Qualitative	Research Methodology
				Education	academic education and to professional training; theoretical knowledge and practical knowledge.	LIS Education
				History	Historical accounts of the field.	History of IS
Subject-based knowledge	Knowledge on the object/phenomenon (i.e., the mediating aspect & conditions of how an knowledge)	2. Resources		Issues	quality information (resources), information (resources) quality	Information Quality
				Types	Primary resources (i.e., the human originators), secondary resources, tertiary resources	Information Systems
		3. Knowledge Workers		Issues	A. Personality traits B. Theoretical knowledge C. Applied knowledge and practice	Information Ethics
				Types	Taxonomies of professional workers by fields of expertise (e.g., medical informatics), and organizational sector (e.g., librarians, archivists)	LIS Education
		4. Contents		Issues	Content related issues (e.g., What is a subject?)	
				Types	Taxonomies of structures (e.g., knowledge maps, subject classification schemes, thesauri), classification systems (e.g., LCC, DDC, UDC, CC, BC), subjects (i.e., Archaeology, biology, Computer Science) and the like.	
		5. Applications		Issues	Issues related to the development of application oriented systems.	
				Types	Taxonomy of applications (e.g., (information) searching, shopping, socialization and socializing).	
		6. Operations & Processes		Issues	Issues related to the various operations and processes involved in mediating human knowledge.	
				Types	Taxonomy of operations and processes: documentation, representation, organization, processing, dissemination, publication, storage, manipulation, evaluation, management, searching, and retrieving knowledge.	
7. Technologies		Issues	Technological related issues (e.g., user-interface design).			
		Types	Taxonomy of knowledge technologies and media: electronic-based technologies (e.g., computer-based information systems, Internet), paper-based and printing-based technologies (e.g., books), communication-based technologies and media (e.g., cellular phones, MP3).			
8. Environments		Issues	Social issues (e.g., information policy, information accessibility), including ethical and cultural issues, professional issues related to the settings, as well as legal issues (e.g., intellectual property, privacy), and ethical issues (e.g., privacy vs. public interests).	Information Ethics		
		Types	A. Ethical & Cultural environments B. Settings (e.g., Education, Health)	Social Informatics		
9. Organizations		Issues	Issues related to the organizational settings (e.g., managing knowledge in business organizations)			
		Types	A. Organizational Type: 1. Governmental Sector 2. Public sector 3. Private sector B. Functional type 1. Memory organizations 2. Information services			
10. Users		Issues	User related issues (e.g., user information needs, user behavior, user search strategies)			
		Types	A. Individuals B. Groups and Communities 1. Gender-based 2. Age-based 3. Culture & ethnicity-based 4. Need & Interest based (e.g., division by profession)	User Studies Information Behavior		

* The words in bold are categories. ** The other terms are exemplary terms (entries).

Figure 7.5: Knowledge map of information science [24]

7.4 Proposed Algorithm for knowledge ETL from Knowledge Maps

Given below is the algorithm proposed for knowledge ETL operations from Knowledge Maps. This algorithm will transform the in Knowledge map into the proposed storage structure for loading into the knowledge warehouse.

A knowledge map can be in many different forms as discussed in the literature review section. This algorithm is only for the Chaim Zins format of hierarchical knowledge maps.

Proposed Algorithm

Create Root node in the forest for this knowledge map

Write decision function to check /Return the type of knowledge map stored in the tree.

For all categories at top most level

 Create Node (Category Text, Root Node)

Next category

//Comment: this function Create Node will take the category and its parent node and will be called recursively for all the sub-categories to form a tree. Recursion is used as the number of levels is unknown

Create Node (Category, Parent Node)

 Create a child node

 Set parent of this child node to Parent Node

 If it is a category node

 Set node type to "Category"

 Write decision function to return the category details

 For all subcategories in this category

Create Node (Subcategory, Child Node)

Next sub category

Else

Set node type to “Description”

Write decision function to return the node description

End if

End function

Limitations of the proposed algorithm

- There are many formats for a knowledge map but this algorithm is for Chaim Zins format of knowledge maps which describes fields in the form of hierarchy of categories and sub-categories.
- There will be a separate tree in the forest for each knowledge map
- The decision tree for knowledge maps will not be a binary tree

7.5 Case study of proposed knowledge ETL algorithm for Knowledge Maps

Another knowledge map developed by Chaim Zins is chosen for case study of this algorithm. The map is about human knowledge and it categorizes different areas of human knowledge in a hierarchical form.

Title of this knowledge map is “10 Pillars of Knowledge” and it is a systematic map of human knowledge [35]. In this map the structure of knowledge and the meaningful relations among the main fields of human knowledge are shown in the form of a chart. According to this map, human knowledge is composed of the following 10 pillars, 1) Foundations, 2) Supernatural, 3) Matter and Energy, 4) Space and earth, 5) non-human organisms, 6) Body and Mind, 7) Society, 8) Thought and Art, 9) Technology, 10) History. The knowledge map is given in the figure below

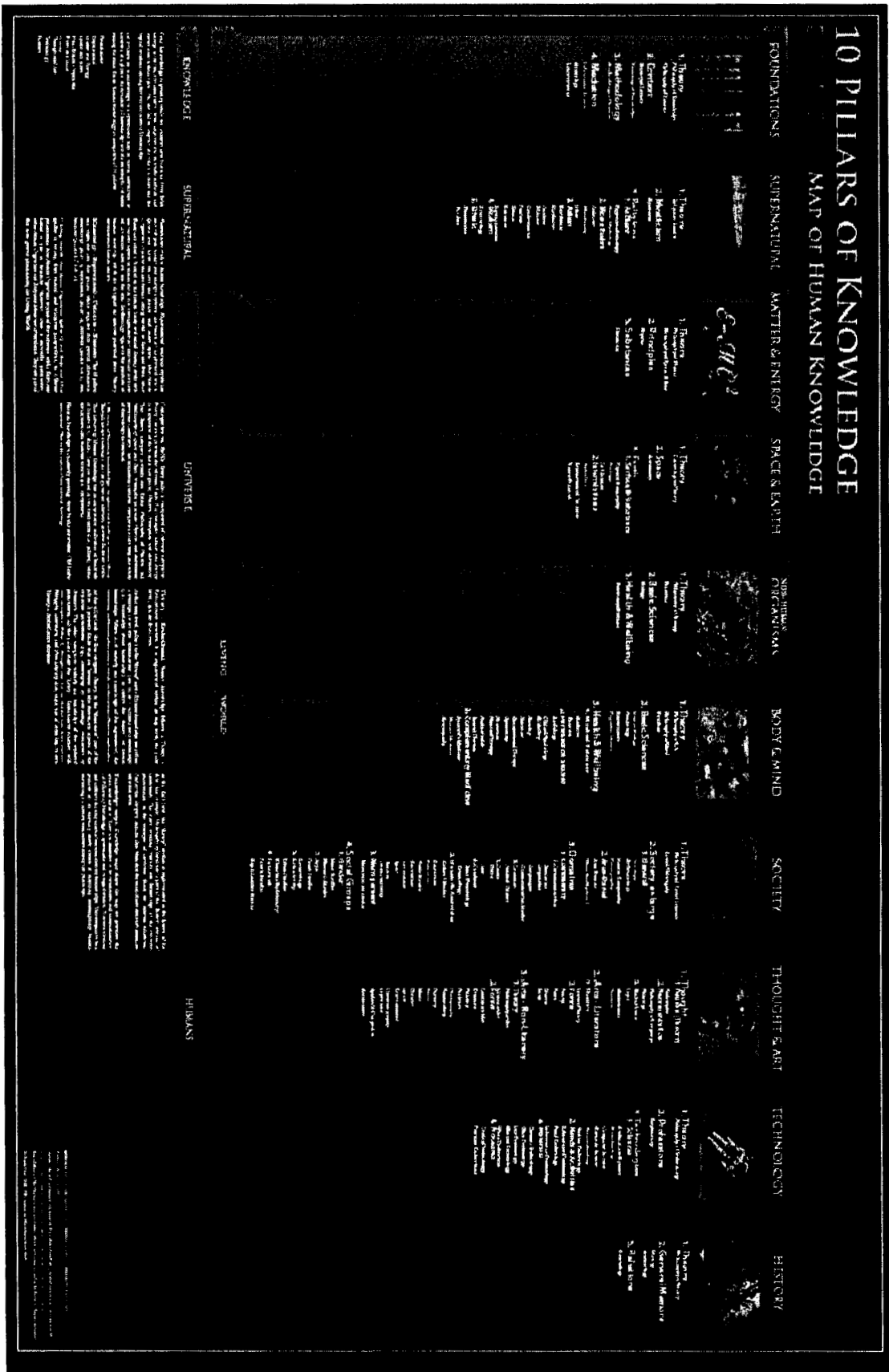


Figure 7.6: Map of Human Knowledge [35]

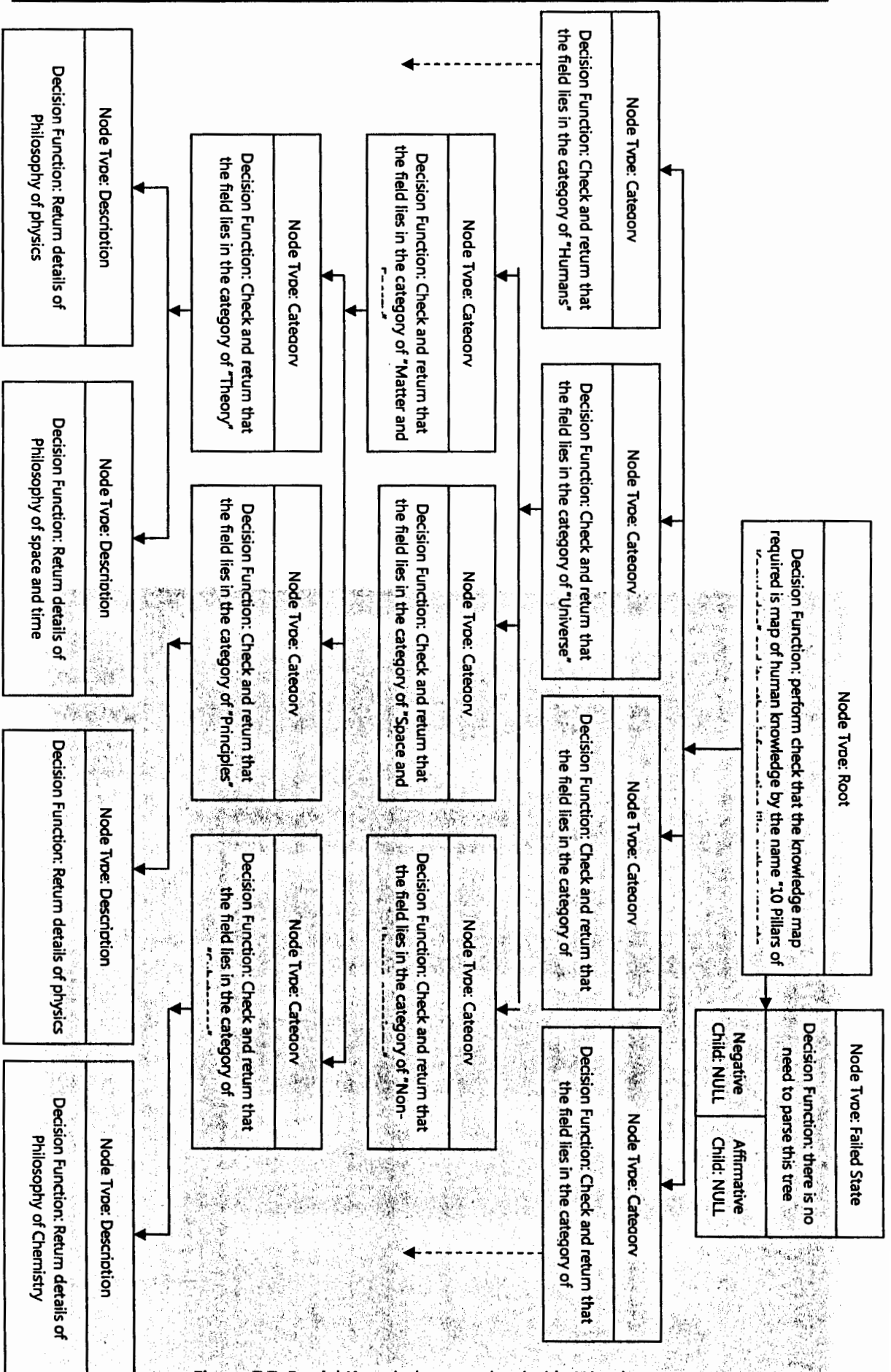


Figure 7.7: Partial Knowledge map loaded in Warehouse

CHAPTER 8



Knowledge Extraction Algorithm for Scripts

8.1 Introduction

Scripts are used for representing procedural knowledge. It represents knowledge as a sequence of events. They were developed by Roger C. Schank and Robert P. Abelson.[18] schank started working on it in 1968 during a study for Artificial intelligence applications. Scripts are playing a vital role in natural language processing. Schank and Abelson used it for a very difficult problem in artificial intelligence which is story understanding [18].

Winston defined a script in 1992 as follows “a script is a remembered precedent, consisting of tightly coupled, expectation-suggesting primitive action and state change frames”. In 1998 Luger and Stubblefield defined a script as “a script is a structured representation describing a stereotyped sequence of events in a particular context”

8.2 Literature Review

Schank Theory

To automate the language understanding, schank came up with the idea that all conceptualizations can be represented in terms of small number of primitive acts that are performed by actors or objects. For example the concept “john read a book” can be represented as “john MTRANS (information) to LTM from book”. Here MTRANS is a primitive act used for mental transfer.

Principles of Schank Theory

The basic principles of schank theory of scripts are as follows

1. Conceptualization is defined as an act or doing something to an object in a direction
2. All conceptualizations can be analyzed in terms of a small number of primitive acts
3. All memory is episodic and organized in terms of scripts
4. Scripts allow individuals to make inferences and hence understand verbal\written discourse.
5. High level expectations are created by goals and plans

Fundamental properties of scripts

To represent natural language stories in form of a script, the script generated must have the following fundamental properties to describe the meanings completely.

- There must be a **lexical** part for representation's vocabulary. It will determine which symbols are being used.
- There must be some constraints on how to arrange these symbols to represent a sentence. This is called the **structural** part.
- A **procedural** part that specifies access procedures that enable you to create descriptions, modify them, and to answer questions using them.
- A **semantic** part that establishes a way to associate meaning with the description.

Properties of a good representation

Following are some basic properties of a good representation of a natural language script.

- You must be able to see what is going on in the script at a glance. The **important objects and relations are made explicit**.
- They show the **natural constraints** between objects. You can clearly express how one object or relation is influencing the other
- They **bring objects and relations together**. Complete sentence is represented together and you can see it in one glance.
- **Irrelevant details are suppressed**. Rarely used details are kept aside but are retrievable when required.
- **Transparency**. You can understand what is being said
- **Completeness**. You can say all that needs to be said
- **Concise**. You can say efficiently all you need to say
- **Fast**. Information can be stored and retrieved rapidly
- **Computable**. They can be created with an existing procedure

Parts of a Script

A script consists of an entry condition, props, roles, tracks and scenes. Entry condition must be satisfied before invoking the script. Props are the objects used in script. Roles are the people involved in a script. Results are the situations occurring after the completion of script. Tracks are the variations that might occur in a script and scenes describe the actual sequence of events in a script [1].

A set of primitive actions has been devised to represent actions in the form of a script. These actions are discussed in the section below.

Primitive actions in a Script

Schank developed a concept of primitive actions to describe the natural languages. He used these primitive actions as building blocks out of which the meanings of different language elements are constructed. He formalized a set of 12 primitive actions and described a sentence in an actor-action-object framework.

A brief description of these primitive actions as described by Schank in [17] is given as under

ATRANS : This action is used to transfer an abstract relationship such as possession, ownership or control. For example “give” is an ATRAN from an actor to a recipient and “take” is an ATRAN from someone to the actor.

PTRANS: This action describes the transfer of a physical location. It requires an actor, object and a direction for example the word “Go” is a PTRAN of an actor by an actor to a location

PROPEL: Describes the application of a physical force to an object. For example “Push” can be described as PROPEL of an object in a direction by an actor

MTRANS: Is used to describe actions like see, tell, read, forget etc. it describes the transfer of information inside an actor, inside the memory.

MBUILD: Describes the construction of new information from old information inside an actor. It is used to describe words like describe, imagine, answer, consider etc.

INGEST: Is used for taking an object by an animal inside himself for example smoking, breathing, eating etc.

GRASP: This is used to grasp an object

ATTEND: It is used to focus a sense organ on an object

SPEAK: To make a noise

MOVE: To move a body part

EXPEL: To push something out the body

PLAN: To decide on a step by step course of action

A script by Schank and Abelson given in 1977 to represent the procedural knowledge of a restaurant is given as under

Props: Tables Menu F - Food Check money	Roles: S - Customer W - Waiter C - Cook M - Cashier O - Owner	Preconditions: S is hungry S has money	Results: S has less money O has more money S is not hungry S is pleased (optional)
--	--	--	--

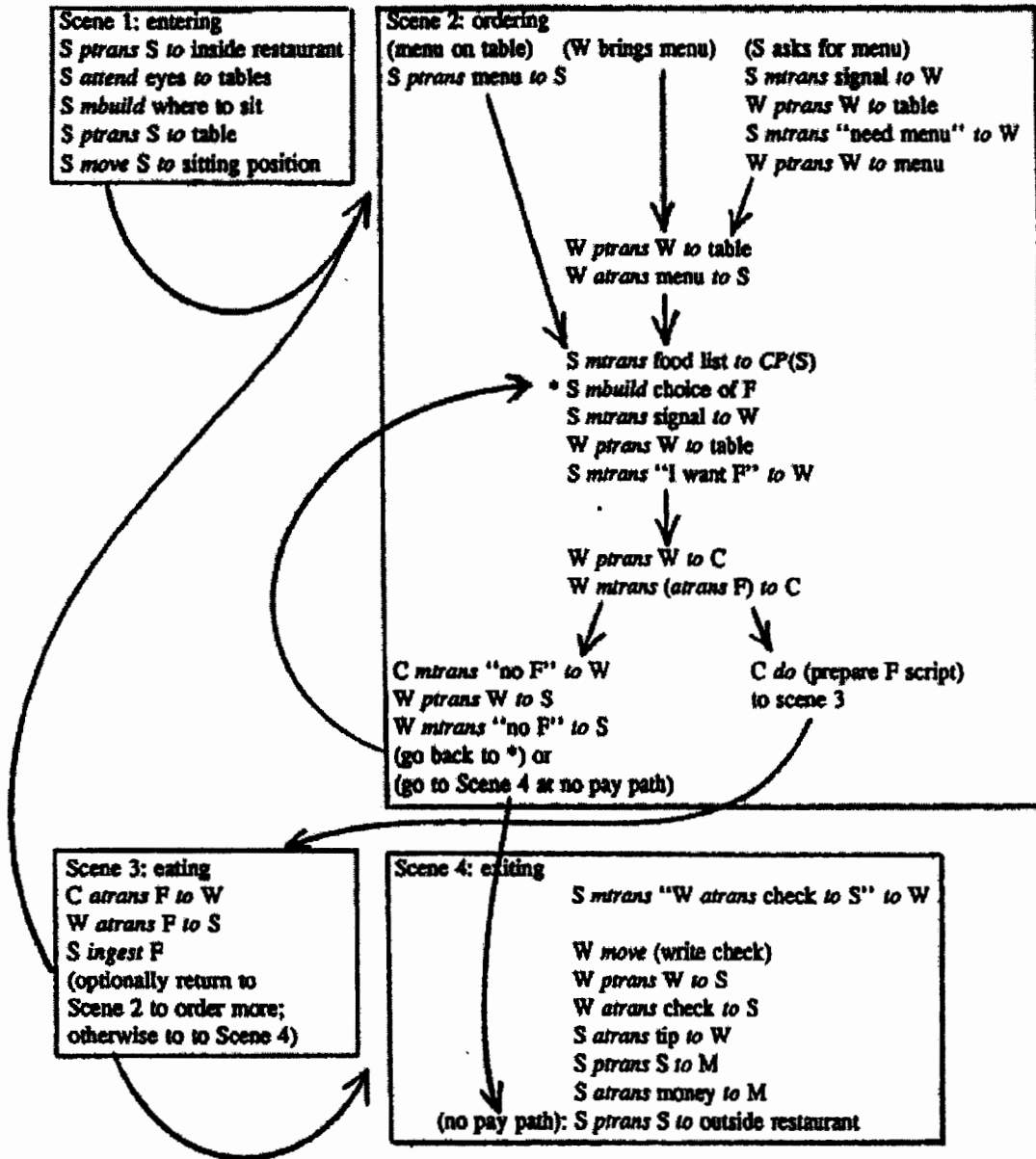


Figure 1 Restaurant script from Schank & Abelson (1977)

Figure 8.1: Restaurant script from schank & Abelson (1977) [15]

8.3 Proposed Algorithm for knowledge ETL operations from Scripts

Create a root node for pre-conditions and write the code to test preconditions

As its negative child node, create a fail state to abort from the system

Start at entry point of scene 1

Next Node: Create a child node

Write code to perform the actions described in the script. If the action is performed by System then write the code to perform that action and if the action is performed by the some actor write code to take input from he actor or else as suitable according to the script and knowledge warehouse purpose.

If any conditions are encountered, write the code to evaluate these conditions and create child notes accordingly

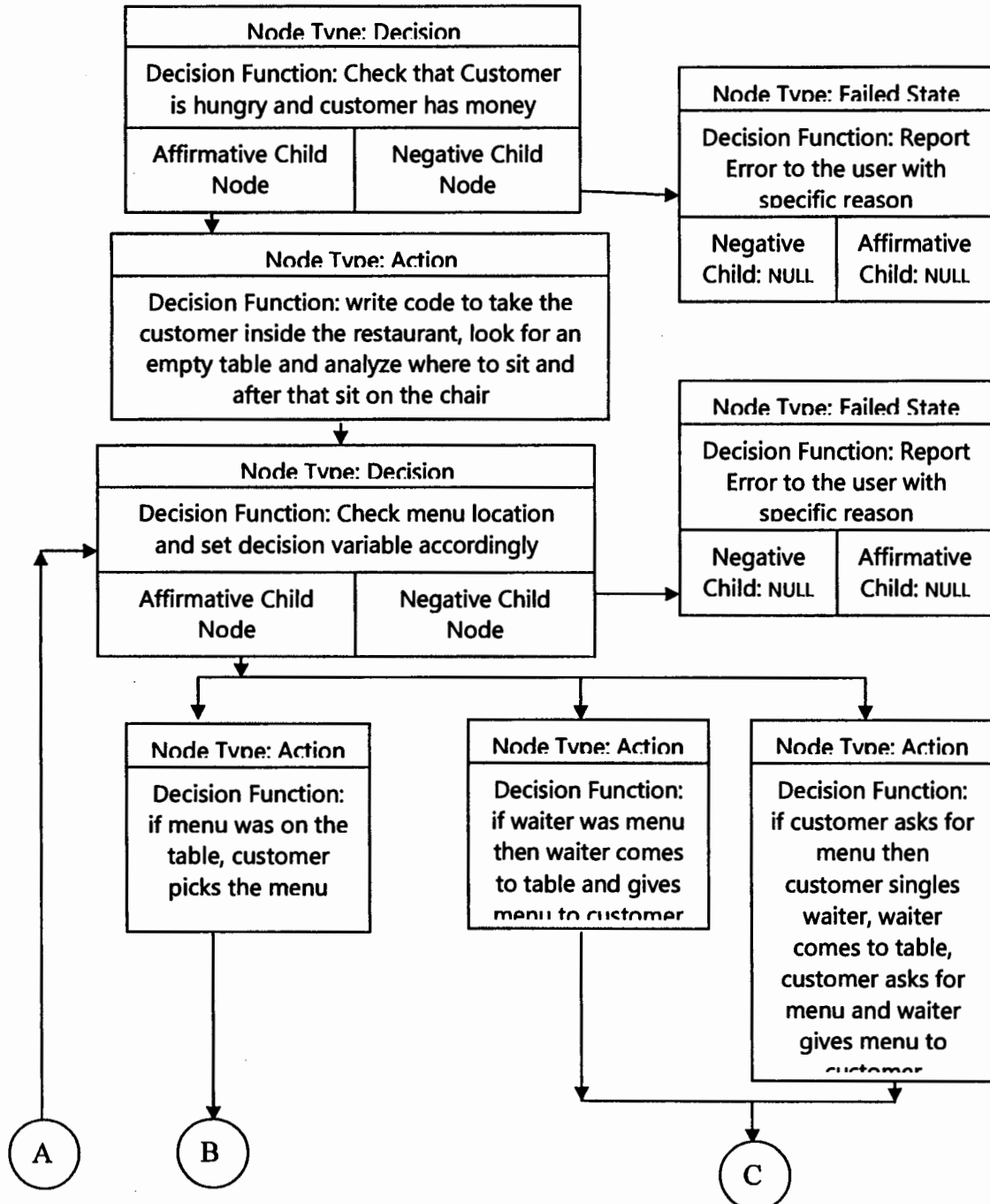
Go to "Next Node" if one of the following conditions is true

1. there is an outgoing arrow
2. role changes from system to external actor or from external actor to system
3. scene changes
4. there is a label with the statement
5. there is a condition

If last scene ends and there is no arrow out then create a node to perform actions in the results and set decision variables accordingly

8.4 Case Study for proposed knowledge ETL algorithm for Scripts

This section will consider the script in figure 9.1 as a case study for the algorithm. A graphical representation of the output of the algorithm which is in the form of proposed knowledge warehouse storage structure is given in the figure below



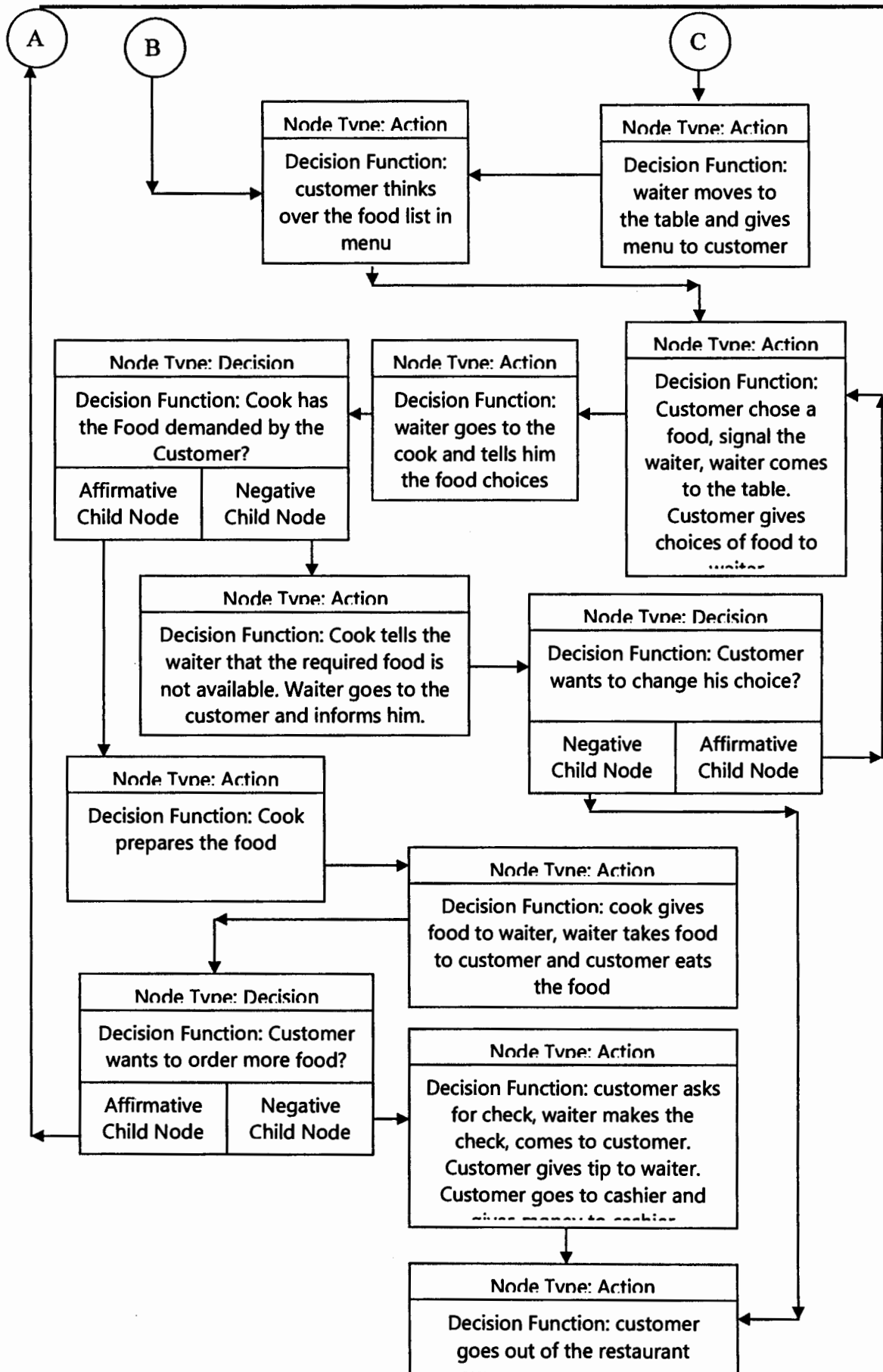


Figure 8.2: Graphical representation of Restaurant script after import in Knowledge warehouse

CHAPTER 9



Conclusion and Outlook

9.1 Introduction to the Outcome of the Thesis

The main focus of research was to take work of nemati and dymond one step further in the field on ETL operations for a knowledge warehouse. the conceptual level framework of a knowledge warehouse and basics of how a knowledge warehouse will function were provided in these research articles.

The work done by nemati and dymond was only a conceptual level work and needs to be refined in focusing on specific areas. This research took the area of ETL operation for a knowledge warehouse and explored it deeper in light of some other research works done on ETL operations for a data warehouse.

The outcome of this research is an ETL framework for knowledge warehouse. it is a comprehensive design focusing on most of the areas of a knowledge warehouse ETL operations.

Structure of a typical node for a knowledge warehouse is also proposed. The basic idea of using a decision tree for knowledge warehouse is proposed by dymond in [2] in a brief way. The node structure was for a knowledge warehouse tree is proposed in this research

Furthermore this research explores the ETL operation of a knowledge warehouse even deeper by proposing knowledge ETL algorithms for 5 types of knowledge documents i.e. decision trees, decision tables, use cases, knowledge maps and scripts.

In brief, this research has explored the area the knowledge ETL in great detail starting from the most abstract in the form of framework and going to the very root level by proposing algorithm for specific knowledge documents.

9.2 Achievements

This research has achieved a lot in its specific area of knowledge ETL from knowledge documents. the achievements of this research are summarized below

1. Knowledge ETL framework:

Although the idea of knowledge warehouse and its advantages were discussed by many authors but the area of ETL operations for a knowledge warehouse was available for a deep research. This research work has proposed a framework for knowledge warehouse ETL operations which can serve a guideline for next researches to come in this specific area. The framework proposed is quite comprehensive as it caters most of the basic needs of ETL operations. Furthermore it is simple.

Knowledge ETL algorithms from specific knowledge sources was also a requirement for this research. This fact required a much deep knowledge of the ETL operation very close to the implementation level. The subject extra research identified many issues that were helpful to improve the framework design.

2. Node Structure for knowledge warehouse decision tree

Almost all the researchers working on knowledge warehouse agreed that the structure of a knowledge warehouse needs to be object oriented. While designing algorithms for this research it was revealed that there is a need to propose a node structure for a knowledge warehouse as this area is yet to be touched in the research. It was also a basic requirement for designing algorithms. Hence a node structure for a knowledge warehouse decision tree is also proposed in this research. This structure is very comprehensive and touches all the traversal, storage and execution requirements of the decision making at node level.

3. Knowledge ETL algorithms from decision trees, decision tables, use cases, knowledge maps and scripts

This research has gone to the root level of knowledge ETL operations for a knowledge warehouse by proposing 5 algorithms from five different types of knowledge documents.

Some algorithms are very specific that even touch the storage of individual elements in the proposed node structure while some algorithms are proposed at abstract level due to diversity in standards of that specific knowledge document.

9.3 Improvements/ Enhancements

A basic structure of a decision tree for a knowledge warehouse is proposed in article of Dymond referenced at [2]. Dymond has proposed a simple structure for a node of a decision tree with a decision variable and a decision function. This node structure is improved/enhanced in this research and a more detailed node structure is proposed.

9.4 Future Work and Recommendations

For this research the recommendations of Nemati and Dymond were followed which suggest that a binary decision tree may be used for storage structure of a knowledge warehouse. During the research it was revealed that a directed graph can be a better option for knowledge warehouse structure.

It was revealed that there is a need to merge knowledge extracted from different sources even from same types of knowledge document for the same purpose. For example there may be 2 different decision trees from different authors for making the same decision. We need to merge all such documents into one structure. Sometimes the rigid structure of a binary decision tree causes difficulties to manage such merging.

In future it is recommended that a directed graph may be used instead of a binary tree. Structure of a directed graph is much more flexible and can be helpful in the tedious and complex task of knowledge ETL.

It is proposed that we can make a “FAILED” and “ACCEPT” states in the graph and all the decisions will end in one of these states.

Furthermore it is proposed that a lookup table can be used as an index for any decision making. Each entry in this table will point to the starting node of the directed graph for that specific decision.

The final structure of a knowledge warehouse will look like the figure below.

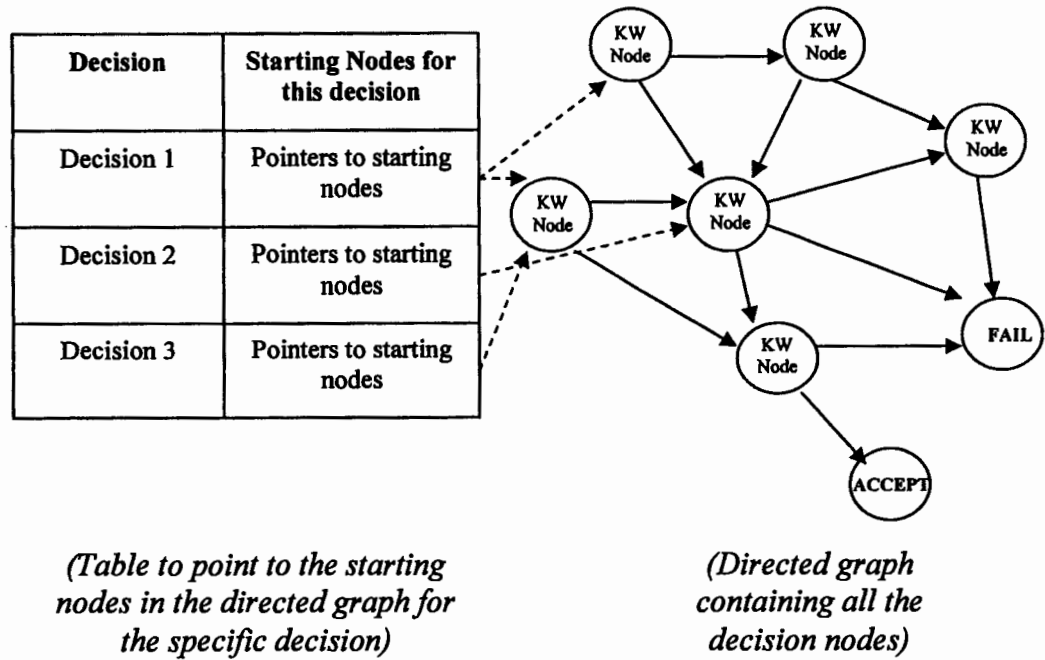


Figure 9.1: knowledge warehouse storage structure in the form of directed graph

Each of the modules in the proposed knowledge ETL framework is a separate topic for future researches.

9.5 Summary of the thesis

The work done in this research is a humble contribution in the field of knowledge ETL for a knowledge warehouse. This work is a mixture of abstract level work and specific in depth work as it touches different areas of the Knowledge ETL process. It proposes frameworks at the abstract level and gets very specific when proposing node structure for a knowledge warehouse and deriving algorithms for knowledge documents.

The research focuses on the area of Knowledge extraction, transformation and loading for a knowledge warehouse. It proposes a comprehensive framework for the knowledge ETL process which has the ability to handle all type knowledge sources and perform the transformations to import them in the knowledge warehouse.

The thesis also encompasses the area of physical storage of a knowledge warehouse by proposing a storage structure and a tree node structure for the knowledge warehouse.

The thesis further goes in detail to the level of proposing algorithms for five knowledge documents and discusses case studies for each algorithm to show how the algorithms will load the knowledge extracted from these knowledge documents and transform them into the proposed knowledge storage structure. The five knowledge documents for this thesis are Decision Trees, Decision Tables, Use Cases, Knowledge Maps and scripts.

On the whole this research has contributed to the domain and also opens new avenues for research.

References

- [1] Efraim Turban, Jay E. Aronson, "*Decision Support Systems and intelligent systems*", 6th edition, Prentice Hall of India, 2004
- [2] Anthony Dymond, "*The Knowledge warehouse: The next step beyond the data warehouse*" Proceedings of the 27th annual SAS Users Group International Conference (SUGI 27), Orlando, Florida, 14-17 April 2002.
- [3] Stephen Golrly, "*Tacit knowledge, tacit knowing or behaving*" Kingston business school, Kingston upon thames, United Kingdom, 2002.
- [4] J'er^ome Darmont, Omar Boussa'id, Jean-Christian Ralaivao and Kamel Aouiche, "*An Architecture Framework For Complex Data Warehouses*", arXiv:0707.1534v1, France, 2007
- [5]: Hamid R. Nemati, David M. Steiger, Lakshmi S. Iyer , Richard T. Herschel, "*Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing*" , Decision Support Systems 33(2002)143-161
- [6] www.wikipedia.org Accessed on 10-Feb-09
- [7] Alex Spokoiny and Yuval Shahar, "*A Knowledge-Based Time-Oriented Active Database Approach for Intelligent Abstraction, Querying and Continuous Monitoring of Clinical Data*", Proceedings of the 11th World Congress on Medical Informatics MEDINFO 2004, M. Fieschi et al. (Eds), Amsterdam: ,2004.
- [8] Panos Vassiliadis Alkis Simitsis Spiros Skiadopoulos, "*Conceptual Modeling for ETL Processes*", ACM, 1-58113-590-4/02/0011, Virginia, USA., 2002
- [9] Jinxin Lin, "*Integration of weighted knowledge bases* ", Toronto, Ont., Canada MSS 1A4, Artificial Intelligence 83 (1996) 363-378, 1996

-
- [10] Michael Yacci, "*The Knowledge Warehouse: Reusing Knowledge Components*", 14623(716) 475-5416.
- [11] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, Manolis Terrovitis, "*A Framework for the Design of ETL Scenarios*", Greece
- [12] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, Manolis Terrovitis, Spiros Skiadopoulos, "*A Generaic and Customizable Framework For the Design of ETL Scenarios* "
- [13] Opim Salim Sitompul and Shahrul Azman Noah, "*A Transformation-oriented Methodology to Knowledge-based Conceptual Data Warehouse Design*" *Journal of Computer Science* 2 (5): 460-465, 2006, ISSN 1549-3636©, 2006
- [14] Thomas Meyer and Kevin Lee, "*Knowledge Integration for Description Logics*"
- [15] Roger C. Schank, "*Script Theory*". <http://tip.psychology.org/schank.html>
- [16] Roger C. Schank, "*Inside Computer Understanding*"
- [17] Roger C. Schank, "*The primitive ACTs of conceptual dependency*"
- [18] [www.wikipedia .com](http://www.wikipedia.com) accessed in June 2009
- [19] G. Ganapathy and V. Rajaraman, "*Information theory applied to the conversion of decision table to computer program*", *ACM Volume 16*, September 1973
- [20] Debasish Kundu, Debasis Samanta, "*An Approach for Assessment of Reliability of the System Using UseCase Model*" *10th international conference on information technology*, 0-7695-3068-0/07, IEEE, 2007

-
- [21] Aziz Salah, "*A use case driven synthesis of state diagram*" Eighth Maghrebian conference on software engineering and artificial intelligence.
- [22] Roger S. Pressman, "*Software Engineering, A practitioner's Approach*", 4th edition, ISBN 0—7-114603-2, Mc Graw-Hill international edition
- [23] www.wikipedia.org accessed on 24th march 2009
- [24] Chaim Zins, "*Knowledge Map of Information Science*" Journal of the American society of information science and technology, 58(4):526-535, 2007
- [25] Mhein An Le Khac, Lamine M. Aouad and M-Tahar Kechadi, "*districuted knowledge map for mining data on grid plateforms*", IJCSNS international journal of computer science and network security, Vol 7. No. 10, October 2007
- [26] J. Havlicek, I. Ticha, "*matrices in knowledge maps*", scientia agriculture bohemica, 39, special issue, 111-116, 2008
- [27] Nhein-An Le-Khac, Lamine Aouad, M-Tahar Kechadi, "*knowledge map: toward a new approach supporting the knowledge management in distributed data mining*", Ireland.
- [28] thian-Huat ong, hsinchun chen, wai-ki sung, bin zhu. "*newsmap: a knowledge map for online news*", decision support systems 39(2005) 538-597, Elsevier.
- [29] Reza saheban, "*knowledge map: Do organizations take advantage of knowledge map?*", Jonkoping 03 2006.
- [30] Milan houska, martina berankova, "*specific type of knowledge map: mathematical model*", kamycka 129, 16521, Prague 6.

-
- [31] lei hong-zhen, zhao peng, zhang me-ling, “*study on the model of knowledge map based on concept clustering*”, china-USA business review, ISSN1537-1514,USA, volume 7, No. 2, Serial no 56, Feb 2008
- [32] I.K. Sethi and B. Chatterjee, “*Conversion of decision tables to efficient sequential testing procedures*”, ACM volume 23, May 1980
- [33] Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich, “*Modern System and Designs*”, 4th edition, Prentice Hall, 2005
- [34] Steve Adolph, Paul Bramble, Alistair Cockburn, Andy Pols, “*Patterns for effective Usecases, the agile software development series*”, ISBN : 0201721848, 9780201721843, Addison-Wesley, 2002
- [35] Chaim Zins, success.co.il/knowledge/index.html , accessed on 28th Aug, 2009.
- [36] Bernard M. E. Moret, “*Decision Trees and Diagrams*”, ACM 0010-4892/82/1200-0593, Computing Surveys, Vol. 14, No. 4, December 1982
- [37] Juhana Salim, Yarina Yahya, Mohd Shahizan Othman, Nurul Rafidza Mohd Rashid, “*The Use of Holistic Approach to Knowledge Management Initiative in Managing Information in Higher Learning Institution: A Perspective*” 6th WSEAS International Conference on E-ACTIVITIES, Tenerife, Spain, December 14-16, 2007.
- [38] Ala’a H. AL-Hamami, Soukaena Hssan Hashem “*An Approach for Facilitating Knowledge Data Warehouse*”, International Journal of Soft Computing Applications, ISSN 1453-2277 Issue 4 (2009), pp.35-40
- [39] Ching-Long Yeh and Jia-Yang Chen “*Building Knowledge Warehouse for Managing the Metadata Layer of the Semantic Web*”, Department of Computer Science and Engineering, Tatung University, Taipei, Taiwan

-
- [40] Azra Shamim, Hameed Hussain, Maqbool Uddin Shaikh, "*A Framework for Generation of Rules from Decision Tree and Decision Table*", International Conference on Information and Emerging Technologies (ICIET) 2010, 6th-9th June 2010, Foundation for Advancement of Science and Technology (FAST), Karachi.
- [41] Galhardas H, (2009) "ETL (Extract-Transform-Load)", https://dspace.ist.utl.pt/bitstream/2295/294968/1/licao_6.pdf accessed on 20-05-2010
- [42] Omer Akgobek , Yavuz Selim Aydin , Ercan ztemel , Mehmet Sabih Aksoy. "*A new algorithm for automatic knowledge acquisition in inductive learning*", *Knowledge-Based System* , 19 :388-395(2006)
- [43] Tasleem uddin, "*Knowledge Warehouse Framework: A new direction towards decision support system for executive management*" Thesis Report, International Islamic University Islamabad, <http://misqe.org/ojs2/index.php/misqe/author/index/completed> (2008)
- [44] Xixu Fu, Hui Wei, "*Cognition Inspired Object Oriented Knowledge Warehouse Architecture*", *Journal Of Software*, Vol 5, No 9 (2010)
- [45] Marcos Zuniga, Francois Bremond, Monique Thonnat, "*Incremental Video Event Learning*", *Springer-Verlag Berlin Heidelberg 2009 , ICVS2009*, LNCS 5815, pp 403-414, 2009
- [46] Mohammad Rifaie, Erwin J. Blas, Abdel Rahman M. Muhsen, Terrance T. H. Mok, Keivan Kianmehr, Reda Alhajj, Mick J. Ridley, "*Data warehouse Architecture for GIS Applications*", In Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS '08) , November 2008, Linz, Austria

-
- [47] Rizwana Irfan ,Dr.Maqbool uddin Shaikh, '*Enhance Knowledge Management Process for Group Decision Making*', International Conference on Computer Engineering and Applications (ICCEA) 2010, Bali Island, Indonesia, 19-21 March, 2010, pp 66 – 70
- [48] Debasish Kundu, Debasis Samanta, '*An Approach for Assessment of Reliability of the System Using UseCase Model*', 10th International Conference on Information Technology, IEEE,7695-3068, 2007
- [49] Saremi A., Esmaeili M., Habibi J., Ghaffari A., "*O2DSS: A Framework for Ontology-Based Decision Support Systems in Pervasive Computing Environment*", *Second Asia International Conference on Modeling & Simulation*, pp 41- 45, 13-15 May 2008
- [50] Dmitry Davidov, Ari Rappoport, "*Extraction and approximation of Numerical attributes from the web*", Proceedings of the 38th Annual Meeting of the association for computational linguistics, pp1308-1317, Sweden, 2010
- [51] Sajidullah, Prof. Dr. Maqbool uddin shaikh, "*Transformation of semantic networks into frames*", accepted for international conference on information and multimedia technology. ICIMT 2010 , to be held on 28-30 December, 2010,. Hongkong , china, sponsored by international association of computer science and information technology (IACSIT) and co-sponsor by IEEE

