# Semi-Automatic Conceptual Data Modeling Using Entity and Relationship Instance Repositories

A Thesis

Submitted to the Faculty

of

Drexel University

by

Ornsiri Thonggoom

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

May 2011

## ACKNOWLEDGEMENT

I am indebted to my supervising professor, Dr. Il-Yeol Song, for his guidance throughout the research program, without which this dissertation would never have been completed. His insightful advice and comments have significantly improved this work. I am very grateful for his unwavering help and support for all these years. And I wholeheartedly thank him for never giving up on me.

I would like to thank the Committee Members, Dr. Don Goelman, Dr. Yuan An, Dr. Harry  Jiannan Wang, and Dr. Xiaohua Tony Hu,  for their valuable advice and helpful comments that contributed to the success of this dissertation.  In particular, I would like to thank Dr.Yuan An for his commenting and reviewing my papers. I would also like to thank Dr. Jiexun Jason Li, who participated in my candidacy exam and offered valuable suggestions.

My success in this program is due largely to the help, friendship and encouragement of many people throughout my graduate career. In particular, Kesarin Phanarangsan has always provided unceasing help and support. Many thanks go to Aunty Ormsin and Uncle Harry Gardiner, Thippaya Chintakovid, and Palakorn Achananuparp for being so generously helpful. My deep appreciation is also extended to Caimei Lu, Ki Jung Li, Namyoun Choi, and all my friends who have made my life here more enjoyable.

Financial support from the Royal Thai Government throughout my study is gratefully acknowledged.

Most importantly, I would like to thank my family for their unconditional love and support. They have confidence in me more than myself. My mother (Manee

Thonggoom) always loves and supports me in everything. My three sisters (Rattapawn, Puttima, and Piyanuch) unfailingly provide cheering-up every time we talk and help me take care of our mother. Without them, I would never have graduated.

# Table of Contents

# LIST OF TABLES

# LIST of FIGURES

# ABSTRACT

**Semi-Automatic Conceptual Data Modeling Using Entity and Relationship Instance Repositories**

Ornsiri Thonggoom
Advisor: Il-Yeol Song, Ph.D

Conceptual modeling is the foundation of analysis and design methodologies for the development of information systems. It is challenging because it requires a clear understanding of an application domain and an ability to translate the requirement specifications into a data model. However, novice designers frequently lack experience and have incomplete knowledge about the application being designed. We propose new types of reusable artifacts called Entity Instance Repository (EIR) and Relationship Instance Repository (RIR), which contain ER (Entity-Relationship) modeling patterns from prior designs and serve as knowledge-based repositories for conceptual modeling. We also select six data modeling rules to check whether they are comprehensive enough in creating quality conceptual models. This research aims to develop effective knowledge-based systems (KBSs) with EIR and RIR. Our proposed artifacts are likely to be useful for conceptual designs in the following aspects: (1) they contain knowledge about a domain; (2) automatic generation of EIR and RIR overcomes a major problem of inefficient manual approaches that depend on experienced modeling designers and domain experts; and (3) they are domain-specific and therefore easier to understand and reuse. Two KBSs were developed in this study: Heuristic-Based Technique (HBT) and Entity Instance Pattern WordNet (EIPW). The goals of this study are (1) to find effective approaches that can improve the novice designers' performance in developing conceptual models by integrating pattern-based technique and various modeling techniques, (2) to evaluate whether those selected six modeling rules are effective in HBT, and (3) to validate whether the proposed KBSs are effective in creating quality conceptual models. To assess the potential of the KBSs to benefit practice, empirical testing was conducted on tasks of different sizes. The empirical results indicate that novice designers' overall performance increased by 30.9~46.0 % when using EIPW, and increased by 33.5~34.9% when using HBT, compared with the cases of no tools.

# 1. INTRODUCTION

## 1.1 Motivations and Problem Statement

Conceptual data modeling is the foundation of analysis and design methodologies for the development of information systems. From academic literature, conceptual design is probably the most critical and important process of developing a quality database application because it not only provides a blueprint for the entire system but also determines most of the system functions and structures (Thalheim, 2000). However, it is difficult because it requires a clear understanding of an application domain and an ability to translate requirement specifications into a data model.

The quality of a conceptual model is measured by the level of accuracy with which it can reflect the real world environment (Dullea, 2003). A good conceptual model has to narrow the gap between real-world concepts and the ability to represent them in a conceptual model. Any errors incurred at this stage could become very costly later after a system has already been implemented. Boehm (Boehm, 1981) reports that the cost difference to correct an error in the early phases of software development as opposed to post-implementation phase is on the order of 1:100. Early in the development of database designs, two qualities, which are performance and storage, have to be measured because of their significant impact on the cost over the life cycle of the database. New hardware technologies have shown that these measurements become less important since their associated costs drop dramatically. However, the database design is the pivotal artifact of the information system. Due to the complexity and size of these systems, the conceptual design phase of the database has still remained extremely important. It is necessary to make it correct, as a poorly designed conceptual data model can lead to a poor database; hence, high costs and poor performance for decades to come.

Natural language (NL) is the common tool for people to describe and communicate their understanding of the world. It has shown that nearly 90% of all the initial requirement specifications in industrial practice are written in NL (L. Mich, Franch, M., Inverardi, P., 2004; Neill, 2003). On the structured representation side, there are many different target formalisms for conceptual data models in different domains. For example, Entity-Relationship diagram (ERD) is often used for database design, while UML class diagram is often used for object-oriented (OO) software design. However, approaches (Batini, 1984; J. Choobineh, Mannino, M., Nunamaker, J., 1988) which skip the NL phase by requiring information analysts to write the requirements directly into formal representations involve much more human effort in the process and cause communication difficulty with the end users. NL was and will remain the main form of requirements documentation (Cheng, 2007).Consequently, in our research, conceptual data modeling is a translating process from NL representations to some kinds of formal representations.

The difficulties in developing the conceptual data models have been stated in many past research studies (Antony, 2002; D. Batra, 2007; D. Batra, Antony, S. , 1994; D. Moody, 2005). These studies show that conceptual designs especially developed by novice designers, lead to unsatisfactory and inaccurate outcomes. Our survey shows that the difficulties in creating conceptual models are:

1. *Ambiguity in a Requirement Specification*

   Most of the inputs in this process are given in NLs which are inherently ambiguous.

2. *Combinatorial Complexity*

   In conceptual modeling, a linear increase in the number of entities can result in a combinatorial increase in the number of possible relationships (D.  Batra, 2007). With merely 5 entities, the numbers of possible relationships are 20 if cardinality is not considered or could be 80 if cardinality is considered.

3. *Semantic Mismatch*

   It represents the inability of novice designers for translating the requirements literally into conceptual modeling structures (D.  Batra, 2007). It is also known that not all real-world relationships match to conceptual modeling relationships; some real-world relationships are derivable at the implementation level. In addition, some real-world relationships become indirect, resulting in ambiguous semantics. Indirect relationships without direct relationships cause semantic ambiguities.

4. *Inexperience and Incomplete Knowledge of Novice Designers*

   Novice designers frequently lack experience and have incomplete knowledge about the application being designed. Even expert designers could fail to obtain a quality conceptual model due to their lack of domain knowledge,  unless they have a clear perception of a requirement specification (N. Kim, Lee, S., Moon, S. , 2008). Concepts that are not explicitly expressed in a requirement are often very difficult to model. Expertise in domain knowledge to identify the hidden entities and relationships is therefore needed (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004).

5. *Scattered Modeling Rules*

   There is no complete set of rules/heuristics that help developing conceptual data models. Also, there is always trade off in design so that not all rules/heuristics can work together because some rules/heuristics are conflicting. These conflicting rules may provide wrong advice.

Currently, there are several commercial graphical CASE tools for automatically converting a conceptual data model into a logical model and into a physical implementation. Most of them offer forward engineering processes, and some of them also reverse engineering processes as well. However, there is still no commercial tool available for translating NL requirement specifications into conceptual data models. At present, a fully automated conceptual modeling approach seems impossible due to the inherent ambiguities in NL, context-dependent nature of modeling, and incompleteness of domain knowledge. It is desirable to develop a semi-automatic process which would be much more economical than an entirely manual modeling process. Therefore, many researchers have proposed knowledge-based systems (KBSs) and tools to support the designers in developing conceptual models (Antony, 2002; J. Choobineh, Lo, A. , 2004; Du, 2008; Harmain, 2003; N. Omar, 2004; Ovemyer, 2001; S. Purao, 1998; S. Purao, Storey, V., Han, T. , 2003). One of the limitations of the proposed tools is that such tools do not solve the problems that novice designers are inexperienced and have incomplete knowledge. In addition, they do not address the semantic mismatch issue.

An ontology can be a source of domain knowledge and designers can use the ontology to get initial domain knowledge. However, developing an extensive and usable domain ontology is labor-intensive and time-consuming (Sugumaran, 2006). Currently, several research projects are considering emerging approaches that try to reuse as much knowledge included in existing large scale ontologies as possible. However, they are general knowledge resources and are not created only specific for conceptual modeling applications. So far there are no good user interfaces or effective APIs to make the process effective and usable (Conesa, 2010). Obviously, domain ontologies are more usable than large scale ontologies.

Most conceptual designs are usually created from scratch, although a similar design might have previously been created. And in many organizations there are a large number of database designs that have been already developed over many years. Reuse of already existing resources and solutions has become a strategy for cost reduction and efficient improvement in the information system development process. Currently, building a library of reusable artifacts involves explication of human developer's knowledge, which is major obstacle in facilitating reuse of knowledge (Ba, 2001; T. Han, Purao, S., Storey,

V. , 2008; Kankanhalli, 2005; Markus, 2001; Orlikowski, 1993). It requires effort from experts to identify elements with potential reuse, and then convert these into reuse elements. One solution to reduce efforts and time of human experts comes from extracting artifacts from prior designs. If this could be conducted for various application domains, then it would assist in creating the practically reusable libraries.

In this research, we explore knowledge-based and pattern-based approaches that help novice designers develop quality conceptual data models. Our methodology also includes database reverse engineering concepts. We propose new types of reusable artifacts that contain knowledge about an application domain, called the entity instance repository (EIR) and the relationship instance repository (RIR), which are repositories of entity instance patterns (EIPs) and relationship instance patterns (RIPs), respectively. An EIP is a pattern of a single entity and its properties. An RIP is a binary relationship with cardinality constraints between two entities. The EIP and RIP can be automatically extracted from prior relational database schemas. Our proposed artifacts are useful for conceptual designs in the following aspects: (1) they contain knowledge about a domain; (2) automatic generation of EIR and RIR overcomes a major problem of inefficient manual approaches; and (3) they are domain-specific and therefore easier to understand and reuse.

Typically, a rule-based approach (P.  Chen, 1983; Harmain, 2003; Hartmann, 2007; N. Omar, Hanna, P., Mc Kevitt, P. , 2004) is a popular technique for conceptual modeling because it could lead designers with known heuristics. However, this approach does not provide an optimal solution to many sophisticate requirements because most of the proposed rules/heuristics were built based on syntax of some specific NLs. These rules cannot overcome the inherent ambiguities of NLs. For instance, entities can be extracted from nouns in English sentences to produce a list of entities. However, the correspondences between entities are not completely perfect since nouns are also used to refer to many concepts that are not usually represented as entities in conceptual models. In general, rules/heuristics are often useful, but sometimes they may lead to cognitive error called bias (D.  Batra, 2007; D. Batra, Antony, S. , 1994; Parson, 2004). To overcome these errors, we should have a small but sufficient set of rules to create quality conceptual models and make them easy to understand.  In this research, we select six data modeling rules termed as the six domain independent modeling rules that are considered a minimum set of rules to teach novice designers in creating quality conceptual models. We evaluate the usefulness of this set of rules by developing a KBS named heuristic-based technique (HBT) that applies these rules to the creation of conceptual data models.

Two knowledge-based systems were developed in this study: HBT and EIPW (Entity Instance Pattern WordNet). The tasks of our KBSs are divided into two subtasks: entity identification and relationship identification. The entity identification processes of our KBSs are different, but the relationship identification process in them is the same. The architectures of each KBS will be presented in later sections.The goals of this study are as follows: (1) to find effective approaches that can improve the novice designers' performance in developing conceptual models by integrating pattern-based technique and various modeling techniques, (2) to evaluate whether those selected six modeling rules are effective, and (3) to validate whether the proposed KBSs are effective in creating quality data models.

## 1.2 Research Questions and Objectives

This research seeks to address the following questions.

***Question 1: Can we automatically create EIR and RIR from the prior relational database schemas?***

Most of the proposed reusable artifacts used for conceptual data models are developed based on a manual approach that is time-consuming and skill-intensive for expert designers. Reusable artifacts for conceptual data models are typically represented in the form of patterns that can be instantiated and combined in different ways to produce concrete design (Blaha, 2010; Castano, 1998; Coad, 1995; Fowler, 1997; T. Han, Purao, S., Storey, V. , 2008). The widely used approach to build such patterns is called domain engineering. A purpose of domain engineering is to create patterns that embody a generic solution to common problems within a specific domain by following a series of predefined steps such as domain analysis, domain design, and domain implementation (Sherif, 2002). With this approach, designers must have very clear knowledge about the specific domain and must identify the boundaries of what objects to be included and what degree they should be abstracted. Therefore, this approach does not seem to be a practically efficient approach because it takes time and effort to develop reusable artifacts. That is a significant drawback of the existing approach that we attempt to address in this research.

In this research, we propose an automatic methodology for creating EIR and RIR, which are the repositories of EIPs and RIPs, respectively. Our methodology includes database reverse engineering concepts. The database reverse engineering approach can provide a solution to the problem as stated previously. It is defined as shown in Figure 1.1, in this research, as the process of examining existing database design schemas to:

(1) Identify database's elements (e.g. entities, attributes, etc.) and their interrelationships.

(2) Determine domain semantics which are not explicitly represented in an application system. Domain semantics are information about the application domain, which should be captured during the requirements specification phase of database design.

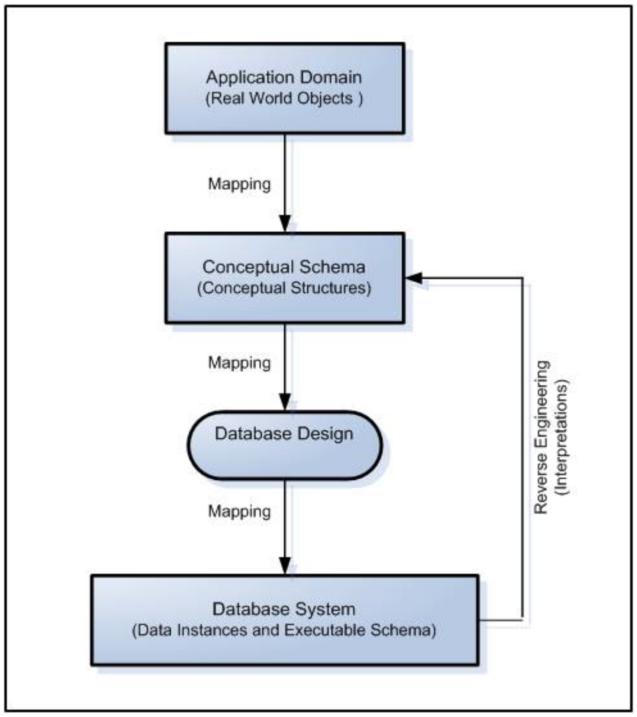


Figure 1.1 The database reverse engineering concept (R. Chiang, Barron, T., Storey, V. , 1994)

The outputs of this database reverse engineering process are transformed into the instance patterns, which are EIPs and RIPs, and can be reused as a source of architecture components for developing conceptual models.

The first objective to address the question above is:

*Objective 1: Developing an automated methodology based on database reverse engineering concept to create the EIR and RIR as new types of reusable artifacts for conceptual model designs.*

The next step of the study is to investigate how EIR and RIR can assist or improve conceptual data modeling. Thus, the next question is:

***Question 2: Can we use EIR and RIR for semi/automatically developing conceptual data models?***

Available commercial graphical CASE tools can help in documenting and analyzing the output of analysis and design. However, they do not provide any support in conceptual data modeling- especially during a stage of identifying the entities, attributes, and relationships, which represent the problem domain. Therefore, many researchers have proposed knowledge-based systems (KBSs) or tools to support the designers in modeling conceptual models. One of the limitations of proposed tools or KBS is that such tools have no domain knowledge or semantic analysis capacity incorporated into them. Therefore, these tools cannot solve the incomplete knowledge of designers and the semantic mismatch issue.

In this research, we explore knowledge-based and pattern-based approaches that help designers develop quality conceptual data models. One of our goals in this study is to provide designers with instance patterns that contain knowledge about an application domain in which the designers are interested. The KBS with EIR and RIR called EIPW (Entity Instance Pattern WordNet) will be developed. The second objective to address this question above is:

*Objective2: Developing a KBS called EIPW with EIR and RIR that contain domain semantics regarding an application domain.*

The following step of this study is to examine whether our proposed EIPW will provide quality conceptual data models and will thus be attractive to the designers. The research question is:

***Question 3: Can the EIPW with EIR and RIR create quality conceptual data models?***

Since our proposed EIR and RIR are automatically extracted from existing relational database schemas, the quality of the KBS with EIR and RIR depends on the scope and correctness of the existing designs. Our research starts creating EIR and RIR based on the library of DDL (Data Definition Language) schemas created by (Silverston, 2001). Later, the instance patterns in EIR and RIR are also extended with case studies. In order to make EIPW more efficient, it is also integrated with other modeling techniques. Although there are several KBSs or tools proposed to assist the designers during conceptual data modeling phase, the efficiency of these systems has not been tested empirically which is a drawback of current research (D. Moody, 2005; Simsion, 2007).

*Objective 3: Evaluating the usefulness of the EIPW by a human subject experiment.*

***Question 4: Are six domain independent modeling rules sufficient to develop quality conceptual model?***

Our survey shows that one of the difficulties in creating conceptual models is the scattered modeling rules. There is no complete set of rules that helps in developing conceptual models. In addition, there are trade- offs in design so that not all rules can work together because some rules are conflicting. We have selected the six domain independent modeling rules based on teaching experiences of over 20 years by one of the committees of this study. These six rules are considered a minimal set of rules to teach novice designers in creating quality conceptual models. These rules are not based on the syntax of any NLs and thus are domain independent. This means that these rules can be applied to a wide range of applications and domains. In this research, we would like to discover if the six rules are indeed useful.  We evaluate the usefulness of these rules by developing a KBS named heuristic-based technique (HBT) that employs these six rules for creating conceptual data models.

*Objective4: Developing the knowledge-based system (KBS) named HBT and evaluating the usefulness with human subjects.*

## 1.3 Contributions

In summary, the contributions of this research are:

1. Proposed an automated methodology for creating EIR and RIR as new types of reusable repositories that contain knowledge about an application domain. The EIR and RIR leverage knowledge from existing designs and serve as knowledge-based repositories for conceptual data modeling by reducing the amount of work on the part of experts.
2. Demonstrated that EIR and RIR are useful for conceptual designs in the following aspects:
    1) They contain knowledge about a domain;

    2) Automatic generation of EIR and RIR overcomes inefficient manual approaches that depend on experienced modeling designers and domain experts;
    3) They are domain-specific and therefore easier to understand and reuse.
3. Evaluated the six domain independent modeling rules as a set of rules that are comprehensive enough to create quality conceptual models.
4. Designed and developed two knowledge-based data modeling tools called HBT and EIPW. HBT incorporates the six domain independent modeling rules, entity categories, and relationship instance repository. EIPW incorporates entity instance repository, entity categories, relationship instance repository, and WordNet. They minimize the cognitive load on designers and ensure that the conceptual models are correct. HBT can serve as a learning tool as well as provide a smooth head-start for novices.
5. Evaluated and demonstrated the utility of the KBSs by human subject experiment.

## 1.4 Dissertation Organization

This dissertation proposal is organized into five chapters. This chapter is an introductory chapter. It lays out research problems expected to be addressed by the proposed research, and gives an overview of our research questions, as well as contributions of the work. Chapter 2 provides a literature review, comprising four major areas. The first area is conceptual data model and discussion of current problems in the field. The second area is techniques of automating conceptual modeling. The tools, systems, frameworks and related work on each technique are surveyed. The third area is database reverse engineering concepts. The fourth area is WordNet as a knowledge-based system for supporting our research methodologies. Chapter 3 describes the research methodologies and research procedures in detail. Chapter 4 presents statistical analysis and empirical results. Chapter 5 summarizes the finding of the study and suggests future work.

# 2. LITERATURE REVIEW

Following the research goals outlined in Chapter 1, this chapter provides a literature review, comprising four major areas. The first area discusses the conceptual data model and current problems in the field. The second area focuses on techniques of automating conceptual modeling. The tools, systems, frameworks and related work on each technique are surveyed. The third area looks at database reverse engineering concepts. The fourth area presents WordNet as a reference system for supporting our research methodologies.

## 2.1 Conceptual Data Models

Conceptual data modeling is the foundation of analysis and design methodologies for the development of information systems. For many years, some researchers have proposed varied conceptual modeling formalisms such as Entity-Relationship (ER) modeling (P. Chen, 1976), derivatives of ER model such as IDEF1X, Oracle CASE notation, and IE, Natural/Nijssen Language Information Analysis Method (NIAM), the Extended Entity Relationship (EER) model, Object Role Modeling (ORM), Object-Oriented (OO) modeling, Unified Modeling Language (UML), EXPRESS, RM-ODP, and others. Thalheim (Thalheim, 2000) estimates that the total numbers of proposed derivatives of ER model are 80. Comparisons of these data modeling formalisms in terms of quality of data models, quality of design time preference, time task performance, and so forth are shown in studies by Kim and March (Y. Kim, March, S. , 1995) and Neill

(Neill, 2003). Of the aforementioned modeling formalisms, EXPRESS, RM-OMP, and UML have been formally designated as standards.

Among the different conceptual modeling formalisms, ER models and UML models are the most widely used in practice (L. Mich, Franch, M., Inverardi, P., 2004). The ER model original proposed Chen (P. Chen, 1976)  has been widely used in structured analysis and conceptual modeling. The ER approach is easy to understand, powerful enough to model real-world problems and readily translated into a database schema (Elmasri, 2004). The ER model consists of a collection of entities, relationships between entities and attributes describing entities and relationships. Many extensions or revisions of ER model have been proposed and utilized in different applications (Gogolla, 1991; Teorey, 1986) such as EER (Extended ER model or Enhanced ER model). The UML is another important conceptual data modeling approach, especially in software engineering. The UML is the standard modeling language for the analysis and design of software. Unlike the ER modeling approach that provides only one type of diagram (ERD), UML 2.2 provides 14 types of diagrams. It is difficult to translate NL requirement specifications to all 14 UML diagrams. Mostly the correspondence between natural language (NL) components and class diagram has been examined in previous research.

A major innovated conceptual modeling formalism is the use of established ontologies as theoretical bases for developing, comparing, and improving data models (Simsion, 2007). Bunge's ontology (Bunge, 1979) has been the most widely used for analyzing conceptual data models (Bodart, 2001; Y. Wand, Storey, V.C., Weber, R., 1999; Y. Wand, Weber, R., 1993, 1995). This ontology has also been adapted by Weber and Wand called Bunge-Wand-Weber (BWW) (Y. Wand, Weber, R., 1988) into a theory of representation that is closer to terminology of the Information Systems community. Some researchers (Milton, 2004) employ Chisholm's ontology (Chisholm, 1996). Both Bunge and Chisholm postulate an objective reality independent of human perceptions. OntoClean (Guarino, 2004) is developed based on philosophical notions for evaluating taxonomical structures. Integration between different ontologies has also been used by different researchers but for different purposes.

Natural language (NL) is the common tool for people to describe things and communicate. Reports (L. Mich, Franch, M., Inverardi, P., 2004; Neill, 2003) show that nearly 90 % of all the requirements in industrial practice are written in NL. Methods that skip the NL understanding phase and that require human analyst to write the requirements directly into formal representations. However, there are at least three limitations for this translation (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004): 1) NL is ambiguous. An effective and accurate analysis is difficult. Therefore, techniques and rules for modeling are required; 2) The same semantic can be represented in different ways. Therefore, ways of handling these style variations are necessary; and 3) Concepts that are not explicitly expressed in a requirement specification are often difficult to model. Expertise in domain knowledge to discover the hidden entities is therefore needed.

## 2.2 The difficulties in creating conceptual data models

The difficulties in creating conceptual data models have been documented in past research studies (Antony, 2002; D. Batra, 2007; Currim, 2008; Dey, 1999; Liao, 2000; D. Moody, 2004; Shoval, 1997). In spite of its importance, research evidences show that conceptual data modeling is not done well, and it should be improved in both training and education (Simsion, 2007). Researchers (D. Batra, Antony, S. , 1994; Currim, 2008; Simsion, 2007) have studied the conceptual data modeling design process employed by novice designers to gain an understanding of errors causing factors. These factors are important in building tools and techniques that can prevent the errors and enhance the quality of information systems. The factors are:

1. *Combinatorial Complexity*

Previous research from a complete survey (Topi, 2002) shows that novice designers have more difficulty in modeling relationships than entities. Batra and Anthony (D. Batra, 2007; D. Batra, Antony, S. , 1994) examine designer performance in modeling open-ended exercises and find that the novice designers not only have difficulty in modeling relationships such as unary and ternary, but also have the difficulty in modeling all kinds of relationships including binary relationships. One of the reasons comes from the fact that given a set of entities, there are potentially a very large number of possible relationships. As the number of entities increases, the number of possible relationships increases at a combinatorial rate as shown in Figure 2.1. What are the rules that allow a designer to choose the right set of relationships in the ER model? And how does the designer know whether all the correct relationships between the entities are identified and which relationships are incorrect? Most relationships can be derived from others, so the problem in modeling relationships is selecting a minimum set that captures the semantics effectively and can be used to derive the others. For the right identification of relationship constructs, the following criteria should at least be met: (1) all semantics in the application should not be lost, (2) all relationship constructs should not be redundant relationships, (3) and the degree of relationship should be minimal.

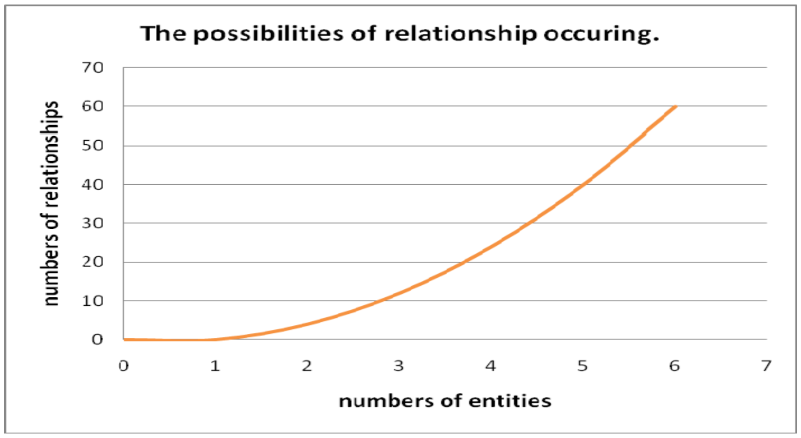


Figure 2.1 The numbers of possible occuring relationships

2. *Scattered Modeling Rules*

There is no complete set of heuristics/rules that help in developing quality data models. In general, heuristics/rules are often useful, but sometimes they may lead to cognitive errors called biases (D. Batra, Antony, S. , 1994; Parson, 2004). Furthermore, there is always trade off in design so that not all the rules can work together because some rules are conflicting. These conflicting rules may provide inaccurate advice.

3. *Semantic Mismatch*

Translating the requirement specification literally into database structures causes literal translation errors (D.  Batra, 2007). For instance, a sentence stating that “an order records a sale of products to customers” may include an erroneous relationship between customer and product.  It shows that not all real-world relationships map to database relationships; some real-world relationships are derivable at the database level. In addition, some real-world relationships become indirect, resulting in ambiguous semantics. Indirect relationships without direct relationships are wrong.

4. *Inexperience of novice designers and incomplete knowledge of designers*

Novice designers have limited knowledge and skills, while expert designers often draw from their past experiences. Even an expert designer might fail to create a quality conceptual model due to their lack of domain knowledge, unless he or she has a clear perception of requirement specifications (N. Kim, Lee, S., Moon, S. , 2008). Expertise in domain knowledge to identify the hidden entities is therefore needed. The important issues are how the novice designers can be trained efficiently and how domain knowledge can be transferred to the designers.

5. *Multiple solutions*

In conceptual design, there is neither a single answer nor an algorithm for creating the best answer. Moody and Shanks (D. Moody, Shanks, G. , 1994) also state that one of the common problem encountered in design is the large number of alternative designs that can be created for a particular problem. Therefore, they propose a six-element framework to evaluate the quality of conceptual data models (D. L. Moody, 1998). Their frameworks are composed of completeness, simplicity, flexibility, understandability, integration, and implement ability. Later this framework is refined, and empirically tested in (D. L. Moody, Shanks, G.G., 2003).

## 2.3 Techniques used for automating conceptual modeling

In the past few years, the field of conceptual data models has spawned numerous techniques for the identification of entities, attributes and their relationships. However, these techniques rely heavily on manual processes and experiences of designers. Currently, there are several commercial graphical CASE tools for automatically converting a conceptual data model into a logical model and into physical implementations such as ERWin, Rational Rose, Visio, Oracle Designer, Dia, etc. Most of them offer forward engineering processes, and some of them reverse engineering processes as well. However, there is still no commercial tool available for translating NL requirement specifications into conceptual data models. At present, a fully automated conceptual modeling approach seems impossible due to the inherent ambiguities in NL, context-dependent nature of modeling, and incompleteness of domain knowledge. It is desirable to develop a semi-automatic process which would be much more economical than an entirely manual modeling process.

This section presents the board scope of techniques used for automatically developing conceptual models. There are at least five categories of techniques used for automatically generating conceptual models design from NL requirement specifications. They are linguistic-based, pattern-based, case-based, ontology-based, and multi-techniques-based.

### 2.3.1 Linguistic-based

Natural language processing (NLP) and linguistic theories are used as a means for designing more user-oriented information and communication systems because NL is a common tool for people to describe and communicate their understanding of the world (Castro, 2009; Métais, 2002). Chen (P.  Chen, 1983)  proposes eleven rules for translating English sentence structures into ER diagram's structure. Since Chen's initial proposal, many studies (Hartmann, 2007; P. Johannesson, Kalman, K. , 1989; N. Omar, Hanna, P., Mc Kevitt, P. , 2004; Ovemyer, 2001) have tried to refine and extend on this approach. However, these rules are still not complete and fully accurate. Although entities can be identified by nouns in a requirements specification, the correspondence between entities and nouns does not completely match because nouns not only refer to entities but also to attributes and other concepts. Entities can also be identified from verb phrases and hidden requirements (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). Most recently, Hartmann and Link (Hartmann, 2007) modify Chen's eleven rules for the translation from English sentence structures and EER elements in which they re-organize and extend those rules in twelve heuristics. However, even these heuristics are not complete.

Therefore, this technique can only serve as a basis for a manual or semi-automatic process of transforming an English specification into ER model. Furthermore, most of the proposed rules are built based on syntax of some specific NLs. These rules cannot overcome the inherent ambiguities of NLs. In addition, most of the languages in this world are very different and, therefore, these kinds of rules can not apply worldwide.

In order to solve the inherent problems of NL and to succeed the machine translation, some studies put constraints on the input by restricting either the vocabulary or the sentence structures (Ambriola & Gervasi, 2006; Osborne & MacNish, 1996; Tjoa, 1993) . With these restrictions, simple linguistic processing such as tagging and chunking can achieve reasonably good results. These also improve the tractability of many difficult problems in NLP such as ambiguity and unknown words.  However, the use of controlled languages has some limitations. It can not apply to existing requirement documents. Furthermore, they are not natural and place burdens on the requirements writers. Several formal specification languages such as Z, Object-Z, VDM, B, and OCL have been also proposed for formal model-based specification. They are very expressive but require knowledge to write a correct formalization. However, they lack completely supporting tools and the use of these tools needs deep knowledge of them in order to write them efficiently. Moreover, these languages have been designed for some specific applications, and their use for different purposes may become awkward and difficult. Other researchers propose dialogue tools that help elicit the NL requirement specification (Buchholz, 1995; N. Kim, Lee, S., Moon, S. , 2008). The main disadvantages of these tools are that they require more interventions by users and it is difficult to use them for large scale batch processing.

Theory of classification and categories has been applied to conceptual data modeling (Larman, 2004; I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). Categories are characterized by the properties shared by their members whereas entities could be classified unambiguously according to their common attributes. It is a widely-used technique in identifying entities and classes. In addition, class categories can be used to spot the missing entities or classes. Also, there are some hidden entities or classes that are not explicitly stated in the requirement documents but can be discovered by applying class categories to domain knowledge (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004).

The trend in this technique orients towards the collaboration with huge linguistic dictionaries (Fellbaum, 1998; Miyoshi, 1996)  and common sense ontologies (Lenat, 1995) . Sometime computers are not skillful because they lack basic knowledge that is obvious for humans. Researchers have long sought to grab this basic knowledge for years.  Linguistic dictionaries not only provide semantic links between concepts such as synonym, antonym, hypernym/hypernym (is-a), and meronym/holonym (part-of) but also syntactical and morphological information. A detailed discussion of relationship types is stated in (V.C. Storey, 1993b). WordNet is the best known linguistic dictionary used for conceptual modeling because it is available and it extends to other languages such as European languages, Spanish, Chinese, and so forth. However, the main drawback of

WordNet is that it does not contain many other important semantic relationships (i.e., no relationship between dish and spoon). Therefore, WordNet++, the extension of WordNet, containing special types of relationships that are not available in WordNet, is proposed in (Dehne, 2001).

#### 2.3.1.1 Tools, systems, and related work

Most of the tools or systems proposed for developing conceptual models follow this technique. They apply NLP to extract the model's constructs from requirements specifications or dialogue sessions with a designer for creating conceptual data models (Buchholz, 1995; Burg, 1998; Du, 2008; Eick & Lockemann, 1985; Harmain, 2003; Meziane & Vadera, 2004; L. Mich & Garigliano, 1999; N. Omar, Hanna, P., Mc Kevitt, P. , 2004; V.C. Storey, 1993a; Tjoa, 1993; F. S. C. Tseng, Chen, A.L.P., Yang, W., 1992). Most of these are involved in extending Chen's original approach. A review of some proposed tools or systems can be found in (Du, 2008). Our paper will present only the recent ones.

LIDA (Ovemyer, 2001) is a semi-automatic text analyzing tool that allows designers to produce class diagrams. It tags parts-of-speech and follows Chen's eleven rules where there is an association of nouns with classes, relationships with verbs and attributes with adjectives and prepositional phrases. However, in requirement specifications, class can also be identified from verbs and hidden requirements (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004).

CIRCLE (Ambriola & Gervasi, 2006) is a web-based comprehensive environment for aiding in NL requirements gathering, elicitation, selection, and validation. A human generated glossary and minimal domain descriptions are added to the original requirements. The actual recognition is performed by a number of MAS (Model, Action, and Substitution) rules. A detailed case study of a fictitious missile control system is provided and various stages of requirements analysis are covered.

COLOR-X (Burg, 1998) is based on WordNet. Its main goal is to facilitate the process of generating conceptual modeling. COLOR-X uses linguistic concepts that are similar to Chen's rules for creating system models that reflect both static and dynamic aspects of the referred system. For example, it relates concepts (constructs) stated in WordNet to OO constructs in which object must be identified from nouns and relationship must be identified from verbs. Later (Dehne, 2001), they revise the method by using WordNet++ instead.

CM-Builder (Harmain, 2003) is a NL-based CASE tool aimed to supporting the conceptual analysis stage of software development in an OO framework. This tool use NLP to analyze the requirement specifications and develop initial UML class diagrams. CM-Builder can work either automatically or interactively with the user. This tool converts all nouns into candidate classes and verbs into relationships. For every candidate class, the frequency in a requirement specification is counted before items are selected. The most frequent candidates are the most likely classes. Attributes are identified from heuristics such as possessive relationships and the use of verb phrase like "to have".

However, this tool has some limitations in its linguistic analysis because NL is ambiguous, fuzzy, and redundant.

ER-Converter (N. Omar, 2004) is a rule-based system. The rules are associated with weights according to the confidence level at which the event is true. The weights assigned to each rule are based on intuition. For example, “If a noun occurs before the verb ‘has/have’, it may indicate an entity type.” With a weight of 0.7, which means that 70% of the time this rule will create the correct result (because not all nouns before the verb ‘has/have’ are entity types). User interventions are required when the calculated weights are low in the processing.

ACDM (Du, 2008) is an automated multi-component system. The system is a fully integrated composite of existing, publicly available components including a parser (Link Parser), a lexical filter (WordNet) and a semantic filter (Google web corpus search facilities). After parsing, it uses extended Chen’s rules to identify ERD’s elements. The main limitation of this system is that the input has to be controlled language requirement specification.

Tseng and Chen (F. Tseng, Chen, C., 2008) propose a translation scheme for transforming NL queries into relational algebra through the class diagram notations. Based on a logical form developed by extending the UML class diagram notations, a transformation model is presented to support the automatic transformation of natural language queries into relational algebra by employing appropriate NLP techniques and OO analysis methods. The proposed logical form has the advantage that it can be mapped from NL constructs by referring to the conceptual schema modeled by class diagrams, and can be efficiently transformed into relational algebra for query execution.

#### 2.3.1.2 Strengths and drawbacks

The domain independence is the strength of this approach. However, the strength of this technique is also its weakness because tools or systems proposed have no domain knowledge incorporated in them. This technique does not provide an optimal solution to many sophisticate requirement specifications because of the nature problems of NL.

### 2.3.2 Pattern-based

The important role of patterns in design is recognized in Alexander's book (Alexander, 1979) on architecture and urban planning. It suggests that designers should produce and use patterns rather than solving problems from the first principle. Now patterns have been well established as a technique for reusing solutions of recurrent problems in the software development process. Pattern reuse provides many benefits such as higher productivity, software quality improvement, and reduction of time and cost for software development. Design patterns have been proven very useful in speeding up the design process through reuse, and in improving the overall quality of systems. Integrating patterns into conceptual design is challenging. Thalheim (Thalheim, 2000) also supports pattern-based modeling in his extensions to the ER model. The recognition of patterns in the context of conceptual data modeling is based on works by Coad et al. (Coad, 1995), Hay (Hay, 1996), and Fowler (Fowler, 1997). Like architects, they create a library of proven structure components, and provide some examples of adapting generic models to suit particular requirements. Empirical research shows that experts reuse patterns while novices do not (Chaiyasut, 1994). The process in pattern reuse can be divided into three tasks: retrieval, adaptation, and integration (Anthony, 2009). Retrieval involves choosing patterns that may be relevant to a particular problem. After a pattern is chosen, it must be adapted or instantiated to fit the specific problem. Finally, it needs to be integrated with other patterns to form a complete model in the form of a conceptual data model. Several authors have proposed various kinds of patterns (Coad, 1995; Fayad, 1997; Fowler, 1997; Gamma, 1995; Hay, 1996; P. Johannesson & Wohed, 1999; Johnson, 1988; Pree, 1994; Silverston, 2001; Szyperski, 1998). For the latest one, Blaha (Blaha, 2010) proposes several types of data modeling patterns: Universal antipatterns are the patterns that we should avoid for all applications; Archetypes are the common modeling patterns occurring across different applications; Canonical patterns are corresponding to meta models of modeling formalisms. However, their utility to conceptual modeling varies greatly. Finally, he presents methods for mapping his patterns to relational schema for database design.  Three criteria (T. Han, Purao, S., Storey, V. , 2008) can be used to examine these patterns. First, usability specifies the ease where an artifact can support retrieval (search and adaptation of the artifact for the current design) and assembly (integration of reusable artifact with other parts of design). Second, reusefulness is measured by the artifact’s granularity and abstractness (domain independence). Third, efficiency is measured by the effort necessary to create the artifacts. Table 2.1 summarizes the quality of each pattern based on the aforesaid criteria.

Table 2.1 Comparing the properties of patterns adapted from (T. Han, 2002)

| Patterns | **Usability** | | **Reusefulness** | | **Efficiency** | |
|---|---|---|---|---|---|---|
| | **Ease of retrieval** | **Ease of assembly** | **granularity** | **abstractness** | **Creation effort** | **Reuse effort** |
| Domain Models (Prieto-Diaz, 1987) | high | N.A. | coarse | medium | high | high |
| Analysis Patterns (Coad, 1995) | medium | medium | small | high | high | high |
| Analysis Patterns (Hay, 1996) | medium | medium | medium | medium | high | high |
| Analysis Patterns (Fowler, 1997) | medium | medium | medium | medium | high | high |
| Analysis Patterns (D. Batra, 2005) | medium | medium | medium | high | high | high |
| Data Modeling Patterns (Blaha, 2010) | medium | medium | small | high | high | high |
| Framework (Fayad, 1997) | medium | medium | coarse | high | high | high |
| Semantic analysis Patterns (Fernandez, 2000) | high | medium | coarse | high | high | high |
| Components (Szyperski, 1998) | low | medium | fine | low | high | high |
| Domain Fragment (T. Han, Purao, S., Storey, V. , 2008) | high | medium | coarse | low | medium | medium |

From the comparison table, the design and construction of prior reusable artifacts are labor-intensive and require a lot of time and effort from expert designers. It is shown that the third criterion, efficiency, is the most daunting obstacle to successful reuse.

Recently, there are packaged data models (or model components) available, which can be purchased and after suitable customization, assembled into full-scale data models. These generic data models are designed to be used by organizations within specific industries. Well-known examples of packaged data models are provided by (Silverston, 2001) and (Kimball, 2002). In summary, the use of packaged data models yield two major advantages to an organization (Hoffer, 2004):

1) Reducing the implementation times and cost.

   The time required to design and implement a large data model can be reduced by weeks or months by using this approach.

2) Providing quality models.

   Packaged data models are created and tested by knowledgeable developers based on their experience with many industries and organizations. They tend to represent best practice data modeling technique.

However, packaged data models cannot replace sound database analysis and design. Skilled analysts and designers are still needed to determine database requirements and to select, modify, and integrate any packaged systems that are used.

#### 2.3.2.1 Tools, systems, and related work

The traditional way of using patterns is to leave the decision of how to adapt or instantiate a pattern entirely up to the person designing the schema. Recently, analysis pattern repositories have been the most popular and employed in conceptual modeling tools or systems. Analysis pattern repository is a group of generic objects with stereotypical properties and relations in a domain neutral manner (D. Batra, 2005). To make use of patterns from a repository, the designer must be able to match a task description with a candidate pattern. According to Structure-Mapping theory (Gentner, 1998), there are three ways comparisons can be made. They are (a) Literal comparison, (b) Abstraction, and (c) Analogy. To reuse analysis patterns, Fernandez and Yuan (Fernandez, 2000) propose an approach that involves looking for semantic analysis patterns that match requirements exactly while trying to specialize analogous or abstract patterns that may apply. However, there are not many proposed conceptual modeling tools or systems using this technique.

APSARA (S. Purao, 1998) is a KBS (knowledge-based system), which automated analysis patterns to create OO conceptual design. It firstly uses NLP to parse the requirement specifications into significant keywords, and eventually objects. Based on the objects identified, analysis patterns are retrieved from the pattern repository, then instantiated and synthesized into a conceptual model. In its pattern repository, thirty analysis patterns by Coad (Coad, 1995). Later, this KBS is  improved  by incorporating learning mechanisms, which provide the designer with additional support by suggesting specific patterns that might apply (S. Purao, Storey, V., Han, T. , 2003). The limitation of this approach is that analysis patterns are so abstract that mismatches to patterns are fairly common. Novice designers seem to have inability to reason with analogy (Anthony, 2009).

Modeling Wizard Tool (Wohed, 2000) is a dialogue tool for selecting the appropriate patterns. Various versions of patterns are stored. The appropriate one is selected step by step according to the answers given to the questions such as "Does the booking consist of one object or may it consist of several objects?" This tool requires much more on user intervention, and it is hard to use it for large scale batch processing.

#### 2.3.2.2 Strengths and drawbacks

Patterns have proven very useful in speeding up the design process through reuse, and in improving the overall quality of systems by promoting the use of designs that have been proven superior in many applications. The advantage of reusable patterns aims not only to reuse schema compounds but also to reuse relationships between objects. However, building a repository of patterns involves explication of human developers' knowledge, which is a major obstacle in facilitating reuse of knowledge. To develop pattern repository, designer must have very clear knowledge about the specific domain and must identify the boundaries of what objects to include and what degree they should be abstracted. It takes a lot of time and effort to create pattern repository. Currently, most of the proposed reusable pattern repositories used for conceptual data models is developed based on a manual approach that is time-consuming and skill-intensive for expert designers. Furthermore, most of the proposed tools in this technique use analysis patterns which require manual matching.

One solution to reduce the effort and time of human experts comes from extracting the pattern artifacts from existing designs (T. Han, Purao, S., Storey, V. , 2008). If this could be done for various application domains, then it would assist in creating the practically reusable pattern artifacts. These patterns are also easier to understand and reuse because they are domain-specific.

### 2.3.3 Case-based

Case-based reasoning is a technology used to develop a KBS known as the case-based system. The basic idea is, given the description of a new problem, retrieving from a case base a similar problem and adapting the retrieval to get the solution. Retrieval mechanisms for reusable artifacts intensively require NLP techniques. And indexing techniques could speed up the retrieval of artifacts. Ambrosio et al. (Ambrosio, Métais, & Meunier, 1995) provide a mechanism for flexible queries. Flexible querying is obtained by the automatic modification of the query statements through the relaxation of query conditions in order to recover concepts within a certain semantic distance according to the semantic relations, i.e. synonyms, hypernym, meronym, and similarity.

#### 2.3.3.1 Tools, systems, and related work

Very few have used case-based reasoning where cases of conceptual models are stored, indexed, and used for future design. We can find only three KBSs that use this technique, which are CSBR (V. C. Storey, Chiang, R., Goldstein, R., Dey, D., Sundaresan, S., 1997), DES-DS (Paek, 1996), and CABSYDD (J. Choobineh, Lo, A. , 2004). A comparison between these KBSs can be found in (J. Choobineh, Lo, A. , 2004).

CABSYDD [64] is a case base reasoning system for database schema design. It comprises of two components: a CBR system and a module that will derive conceptual design from first principles. The case indexing used is similar to that used by (Paek, 1996) in which each schema design is hierarchically organized by business area. The hierarchy is organized by categorizing cases using a four tiered structure (sector, subsector, industry group, and department) based on the North American Industry Classification System (NAICS). Case representation included schemas expressed by EER models, textual identifiers for the business area classification, and a textual case description. Matching is performed by calculating the case with the highest matching index score. If no matching cases exist, the system invokes the module to create a new schema design from first principles

#### 2.3.3.2 Strengths and drawbacks

This technique involves storing conceptual models of a large number of applications and providing a key word mechanism that enable users to search for a conceptual model that is a candidate solution for a problem statement. It takes advantage of reusing the previous design. The limitations in this technique are that if any adjustment is required in the conceptual model, it has to resort to the generic data modeling approach. Moreover, adjustments are always required in order to be appropriate for the particular requirement specification. The major disadvantage of this technique is that developing the conceptual model libraries and indexing mechanism are very expensive.

### 2.3.4 Ontology-based

Ontologies have been proposed as an important way to represent real world knowledge and, at some level, to support interoperability (Soares & Fonseca, 2007). Research on creating and using ontologies has been motivated by the Semantic Web and knowledge reuse. Ontology can range in expressivity from a taxonomy (a parent-child structure), to a thesaurus, to a domain model, to a logical theory (very general, consistent and meaningful knowledge), etc. Some papers point out some similarities and differences between ontologies and conceptual data models (El-Ghalayini, 2006; Fonseca, 2007). According to Fonseca (Fonseca, 2007), two criteria that differentiate ontologies from conceptual data models are (1) the objective of modeling and (2) objects to model. Embley (Embley, 2004) suggests that ontology is the key for solving the semantic problems of information systems.

In the conceptual modeling field, many researchers employ ontology for evaluating, improving or developing the conceptual modeling formalisms. Storey (V. C. STOREY, 2005) proposes an ontology to classify the verb phrases of relationships based on research in linguistics and semantic data models. Wand et al. (Y. Wand, Storey, V.C., Weber, R., 1999) propose rules as a theory of constructing the relationships in conceptual modeling practice. Evermann and Wand (Evermann, 2001) examine the mapping between ontological elements and UML elements and propose guidelines on how to use UML elements to model real-world systems in particular. Purao and Storey (S. Purao, Storey, V. C. , 2005) propose a multilayered ontology for classifying relationships by using data abstractions and by separating domain-dependent and domain-independent aspects of the relationship constructs.

However, the major advantage of using ontology for conceptual modeling is the reusability of knowledge repository. This can be developed into two levels: domain ontology and large scale or upper level ontology. Domain ontology (Conesa, 2010) represents concepts, relationships between concepts, and inference rules for a particular domain. The most well-known domain ontologies are the DAML ontologies (www.daml.org), which are created particularly for Semantic Web and contained a repository of more than 200 ontologies. Several tools for creating and querying domain ontologies are available such as Protégé, OWL, SPARQL, etc.  Detail comparison of each tool is shown in (Corcho, 2003). Instead of developing individual ontologies, there has been interest in creating upper level or large scale ontology. Upper level ontology (Conesa, 2010) represents general concepts that are the same across all domains and always consist of a hierarchy of entities and rules that describe those general entities which do not belong to a specific problem domain.  Examples of upper level ontologies are Cyc, ResearchCyc, BFO (Basic Formal Ontology), DOLCE, SUMO, geneontology, etc. For review  and comparison of upper level ontologies see (Mascardi, 2007). The potential usefulness of upper level ontologies lies in the fact that they are domain independent. However, it is difficult to integrate them and make them truly useful. A major problem with existing upper level ontologies is the lack of good user interface and a good API. For example, Cyc is not an ontology of word sense like WordNet. As a result, there is no comprehensive mapping of Cyc concepts into words of NL (Conesa, 2007). Without adequate tool support, it is difficult to work with them. Obviously, domain ontologies are more usable than large scale ontolgies (Conesa, 2010).

#### 2.3.4.1 Tools, systems, and related work

Ontologies have been considered as important components in many applications. Some generic ontologies such as WordNet and Cyc are available, but most applications require a specific domain ontology to describe concepts and relations in the domain. This approach holds the potential of promoting information reused. Currrently, several projects are considering the emerging approaches that attempt to reuse as much as possible of the knowledge included in existing ontologies.

NL-OOPS (L. Mich, Garigliano, R., 2002 ) is an NL-based system. It is based on LOLITA (Large scale Object-based Language Interactor, Translator and Analyser) NLP system, which includes all the functions for analysis of NL: morphology, parsing (1500-rules grammar), semantic and pragmatic analysis, inference, and generation. The knowledge base of the system consists of SemNet, which is a large graph that holds knowledge that can be accessed and expanded using NL input and has been merged with WordNet. Thus LOLITA is among the largest implemented NLP systems. Documents in

English are analyzed by LOLITA and their content is stored in its knowledge base, adding new nodes to its semantic network. NL-OOPS prototype implements an algorithm for the extraction of classes and associations from the semantic network of LOLITA. NL-OOPS's interface consists of three frames. The first one contains the text being analyzed and the second frame gives a partial representation of the SemNet structures used by LOLITA for the analysis of the document. After running the modeling module, the third frame gives a class model.

OMDDE (Sugumaran, 2006) uses a domain ontology to represent the domain knowledge in order to assist database designer. OMDDE is a prototype that demonstrates the possibility of using an ontology as the domain knowledge. The prototype can assist Database designers to design a conceptual model from scratch and also evaluate the existing conceptual database model. The prototype uses an auction domain ontology as the knowledge domain to identify the entities and relationships for the conceptual model. The prototype also suggests terms and highlight missing constructs. Furthermore, it expands the relationship element to include four types of domain relationships: prerequisite, mutually inclusive, mutually exclusive, and temporal.

#### 2.3.4.2 Strengths and drawbacks

Ontology can be the source of domain knowledge and designers can use ontology to get initial domain knowledge. Corcho and his colleagues (Corcho, 2003) suggest that a strategy for developing ontologies would be to reuse large scale or upper level ontologies to create domain ontologies or knowledge bases. The same upper level ontology can be used for developing many knowledge bases or ontologies, which share the same skeleton. Extensions of the skeleton should make at the low level by adding domain-specific subconcepts.

However, ontology development is fundamentally difficult. For example, Cyc takes more than 20 years and 900 people to develop a large scale common sense knowledge base. Even for a specific domain, developing intensive domain ontology requires labor intensive and time-consuming. Automatic ontology development is difficult work because of the lack of structured knowledge base or domain thesaurus. While many ontology tools such as OntoEdit, Ontolingua, and Protégé are available to aid the development of ontologies, ontology constructions still need human effort. Most studies of ontology development and application assume manual process.

### 2.3.5 Muti-Techniques-Based

From our survey, most tools or systems for conceptual design require users' involvement during the process. And no single technique works best all the time because each technique has some limitations. Ideally, various techniques should be integrated together for a design process. For example, Song et al. (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004) have proposed a TCM (Taxonomic Class Modeling) methodology used for object-oriented analysis in business applications. This method integrates several class modeling techniques under one framework. Their framework integrates the noun analysis method, class categories, English structures, check lists, and modeling rules. Thonggoom and colleagues (Thonggoom, 2011) propose EIPW (Entity Instance Pattern WordNet), which is a knowledge-based database modeling tool. It integrates pattern-based technique and various modeling techniques. This tool shows how domain knowledge stored in the instance patterns can be used together with other modeling techniques.

## 2.4 Evaluation Method

Evaluation acts as a significant after-stage in all surveyed techniques. Although there are several systems or tools proposed to assist the designers during conceptual modeling phase as shown in Table 2.2, the efficiency of these systems has not been tested empirically which is a drawback of current research (D. Moody, 2005; Simsion, 2007). Also, in this field, there are no universal standard guidelines for measuring the performance of the proposed tools or systems. Some researchers use individual grading frameworks for the evaluation, while some use recall and precision as the measurements for evaluation. Even though they use the same method for the measurement, the conditions of measurements are very different. Since there is no standard evaluation of NL-based tools available, it is difficult to compare the performances between and among the tools.

Table 2.2 A summary of conceptual modeling tools or systems

| | **Input** | **Output** | **Techniques used** | **User Intervention** | **Lexical Knowledge** | **Domain Knowledge** |
|---|---|---|---|---|---|---|
| ACDM (2008) | Controlled NL text | ER model | Linguistic Rules, Lexical Filter | Can be automatic or semi-automatic | Yes | No |
| APSARA (2003) | NL text | ER model | Analysis Patterns, Learning Concepts | Yes | No | No |
| CABSYDD (2004) | NL text | EER model | Case-based Reasoning | Yes | No | Yes |
| CIRCLE (2006) | NL text | View model, Validated requirement | Linguistic Rules, Requirement Validation | Yes | No | Yes |
| CM-Builder (2003) | NL text | Class diagram | Linguistic Rules, Frequency Analysis | Can be automatic or semi-automatic | Yes | No |
| Color-X (1998) | NL text | OO constructs | Linguistic Rules, Paraphrase Dialog | Yes | Yes | No |
| EIPW (2011) | NL text | EER model | Instance Patterns, Entity Categories, Modeling Rules | Yes | Yes | Yes |
| ER-Converter (N. Omar, 2004) | NL text | ER model | Linguistic rules | Can be automatic or semi-automatic | No | No |
| LIDA (Ovemyer, 2001) | NL text | Class diagram | Linguistic rules | Yes | No | No |
| Modeling Wizard Tool (2000) | NL text | EER model | Analysis Patterns, Dialog Sessions | Yes | No | Yes |
| NL-OOPS (2002) | NL text | Object model, Revised requirement | Linguistic Rules | Yes | Yes | Yes |
| OMDDE (2008) | NL text | ER model | Domain Ontology | Yes | No | Yes |
| TCM (2004) | NL text | Class diagram | Noun Analysis, Class Categories, Modeling Rules | Yes | No | Yes |

## 2.5 Database Reverse Engineering (DBRE)

DBRE is another relevant area of our research. Typically, the database design process can be defined as a sequence of schema transformations that convert the user requirements into an executable schema expressed by the DDL (data definition language) of the target DBMS (H. Chiang, 1993). The entire process can be written as a function:

DDL schema = Database Design

On the other hand, database reverse engineering (DBRE) is the reverse of the task of database design. According to Chiang et al. (R. Chiang, Barron, T., Storey, V. , 1994), DBRE is the process of recovering such a conceptual data model by examining an existing database system to identify the database's elements and their interrelationships. It aims at extracting a conceptual data model from a relational database schema. The entire process can be written as a function:

Conceptual schema = DBRE (DDL schema, data instances)

Methods of DBRE have been proposed since 1980s. Dumpala and Arora (Dumpala, 1981) were the first to focus on DBRE field. Fahrner and Vossen (Fahrner, 1995) provide a survey of various methods to reverse schema transformations from the relational schema into the ER model according to five characteristics. These characteristics are: (1) the various extensions of the ER model, (2) transformation prerequisites, (3) the principle transformation method, (4) properties of transformation methods, and (5) user interaction requirements. Most of the DBRE methods we have reviewed are informal. In particular, they depend on various rules/heuristics to generate elements in a conceptual model from available sources and do not formally specify the quality of the results. DBRE is also difficult to automate and requires human intervention. Since the sources do not contain sufficient semantic information, the conceptual models created by DBRE methods are often closely tied to the existing database schemas and so may become just the graphical representations of the actual logical and physical implementations of the databases. The methodology proposed by Chiang (H. Chiang, 1993) divides the reverse engineering process into six phases: (1) Initialization, (2) Decomposition, (3) Classification, (4) Generalization, (5) Identification, and (6) Refinement & Enhancement. Figure 2.2 shows all these phases and the functions of each phase.

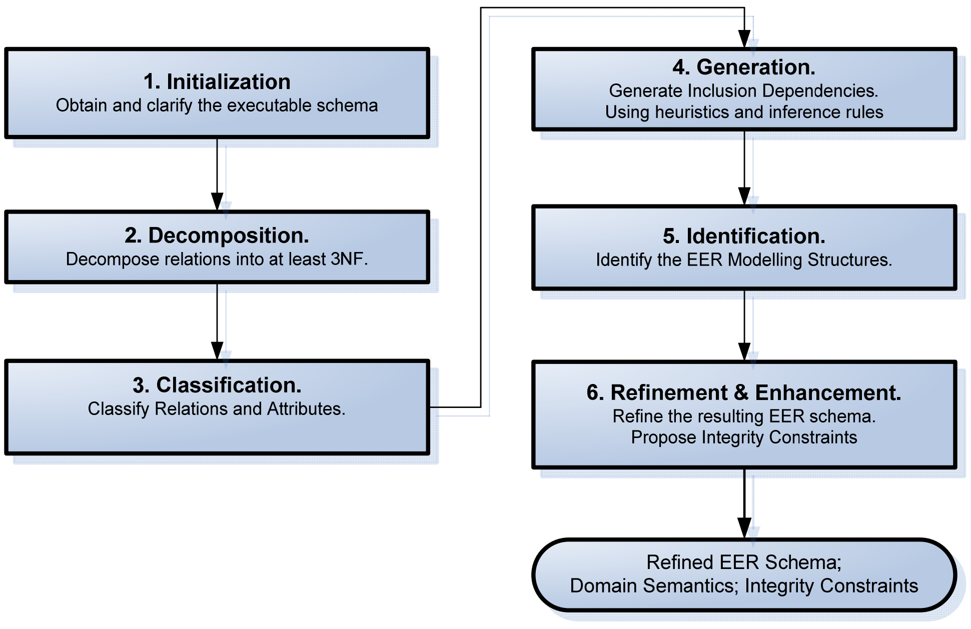

1. Initialization
Obtain and clarify the executable schema
2. Decomposition.
Decompose relations into at least 3NF.
3. Classification.
Classify Relations and Attributes.
4. Generation.
Generate Inclusion Dependencies.
Using heuristics and inference rules
5. Identification.
Identify the EER Modelling Structures.
6. Refinement & Enhancement.
Refine the resulting EER schema.
Propose Integrity Constraints
Refined EER Schema;
Domain Semantics; Integrity Constraints

Figure 2.2 Database Reverse Engineering Process (H. Chiang, 1993)

In our research, we employ the DBRE methodology proposed by Chiang (H. Chiang, 1993) for automatically developing the proposed reusable pattern repositories.

## 2.6 WordNet

WordNet (Fellbaum, 1998) is an online lexical reference system. It was created and improved at Princeton University since1985. It groups words into a set of synonyms called synset, and maintains the various semantic relationships between these synonym sets. The latest version of WordNet is 3.0, which contains about 155, 000 words organized in over 117, 000. The semantic relationships between synsets are IS-A (hypernym/hyponym), Part-of (meronym/holonym), synonym, and antonym. The IS-A relationship is the most fundamental by producing a taxonomic hierarchy of synsets.

WordNet has analyzed large corpora and gathered statistics on the senses in which words are used. For example, the synonym sets for each sense are ranked by frequency. WordNet aims to build a combination of a dictionary and a thesaurus, and to support automatic text analysis and artificial intelligence applications. Due to ambiguities in NL, words may have several meanings (homonyms) and many concepts can be represented by two or more words (synonyms). WordNet has been used as a reference tool to disambiguate nouns in automated conceptual data modeling (Du, 2008). Although entities can been identified by nouns in a problem statement, the correspondence between entities and nouns is not completely matching because nouns do not only refer to entities but also to attributes and other concepts. In fact, it is difficult to automatically identify which nouns should be entities and which should not. Such distinctions depend heavily on context and human ability to apply their own knowledge.

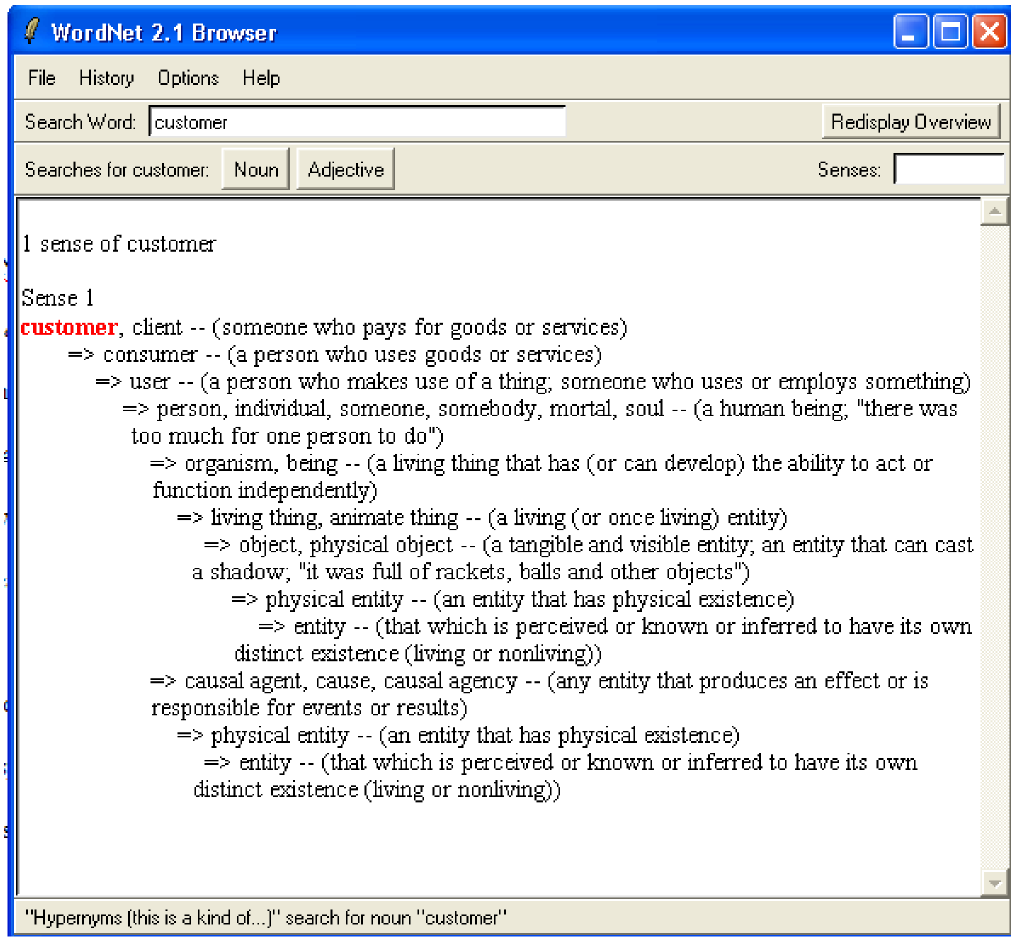
In the conceptual data modeling field, WordNet can be used as a source of reusable knowledge to ensure that the designing models are correct. Métais (Métais, 2002) investigates the use of NLP techniques in the design phase of information systems. In summary, the possible usages of WordNet applying in automated conceptual modeling process are:

- To differentiate between entities and non-entities. The top noun categories and the hypernym chains in WordNet can be used as a general standard to distinguish attributes from entities (Du, 2008). Three top level category groups are defined as follows:

  1) strong-entity: “group”, “physical object”, “physical entity”, “thing”
  2) mid-entity: “substance”, “event”, “communication”, “physical process”
  3) weak-entity: “cognition”, “attribute”, “measure”, “constituent”, “language unit”

  If the hypernym chain of a noun phrase reaches to one of the categories in the strong-entity group, it means that this noun has a high potential to be a candidate entity. On the other hand, if the hypernym chain of the term reaches to one of the categories in the weak-entity group, it means that this noun has a low potential to be a candidate entity.

- For example, the WordNet hypernym chains of the word “customer” as shown in Figure 2.3 are consumer => user => person => organism => living thing => object, while specific nouns that more usually indicative of entity will link to synsets such as “object and physical entity.



```
1 sense of customer

Sense 1
customer, client -- (someone who pays for goods or services)
    => consumer -- (a person who uses goods or services)
       => user -- (a person who makes use of a thing; someone who uses or employs something)
          => person, individual, someone, somebody, mortal, soul -- (a human being; "there was
           too much for one person to do")
             => organism, being -- (a living thing that has (or can develop) the ability to act or
             function independently)
                => living thing, animate thing -- (a living (or once living) entity)
                   => object, physical object -- (a tangible and visible entity; an entity that can cast
                   a shadow; "it was full of rackets, balls and other objects")
                      => physical entity -- (an entity that has physical existence)
                         => entity -- (that which is perceived or known or inferred to have its own
                         distinct existence (living or nonliving))
             => causal agent, cause, causal agency -- (any entity that produces an effect or is
             responsible for events or results)
                => physical entity -- (an entity that has physical existence)
                   => entity -- (that which is perceived or known or inferred to have its own
                   distinct existence (living or nonliving))

"Hypernyms (this is a kind of...)" search for noun "customer"
```

Figure 2.3 An example of WordNet hypernym chains

- To disambiguate the meaning of noun or verb by examining synonyms. For instance, E-R generator (Gomez, 1999) employs WordNet for word sense disambiguation in their conceptual data modeling system. An interface is used to access WordNet and displays the ontological categories for a given word. When word ambiguity happens, the system asks the user to choose the proper category in the current context.

- To discover some hidden relationships through WordNet. The hypernym chains in WordNet can be used to identify the inheritance and aggregation relationships.
- To identify hidden attributes. In CM-Builder (Harmain, 2003), it uses WordNet to assist users in determining the meaning and the context of words, and to identify hidden attributes that may get from adjectives.

# 3. RESEARCH METHODOLOGY

This Chapter presents four methodologies to answer the research questions stated in Chapter 1 during the course of our work:

*3.1 Developing EIR and RIR as new types of reusable pattern artifacts for conceptual model designs.*

*3.2 Proposing six domain independent modeling rules.*

*3.3 Developing EIPW with EIR and RIR that contain domain semantics regarding an application domain.*

*3.4 Developing HBT with six domain independent modeling rules and RIR.*

In our research, we use the ER model originally developed by Chen (P. Chen, 1976) as our representation because it has been widely used in conceptual modeling field -- powerful to real-world problems and readily translated into a database schema (Teorey, 1986).

## 3.1 Developing EIR and RIR

This section presents our automatic methodology for creating Entity Instance Repository (EIR) and (RIR), which are the repositories of Entity Instance Patterns (EIPs) and Relationship Instance Patterns (RIPs), respectively. EIR and RIR contain ER modeling patterns from prior designs and serve as knowledge-based repositories for conceptual modeling. An EIP is a pattern of a single entity and its properties. An RIP is a binary relationship with cardinality constraints between two entities. Examples of these are shown in Figure 3.1. We propose a method based on database reverse engineering concepts (R. Chiang, Barron, T., Storey, V. , 1994) to automatically extract EIPs and RIPs from relational schemas. This methodology employs three assumptions about the characteristics of the input schemas for database reverse engineering processes:

(1) Relational schemas: An input is a DDL (Data Definition Language) schema that contains data instances of an application domain.

(2) 3NF relations: There are no non-3NF relations in the input relational schemas. It would simplify the extraction process by dealing with the relations, each of which primarily corresponds to one entity type or one relationship type, rather than corresponding to more than one entity type or a mixture of entity and relationship types.

(3) Proper primary keys (PK) and foreign keys (FK): Proper PKs and FKs are specified in input DDL schemas.

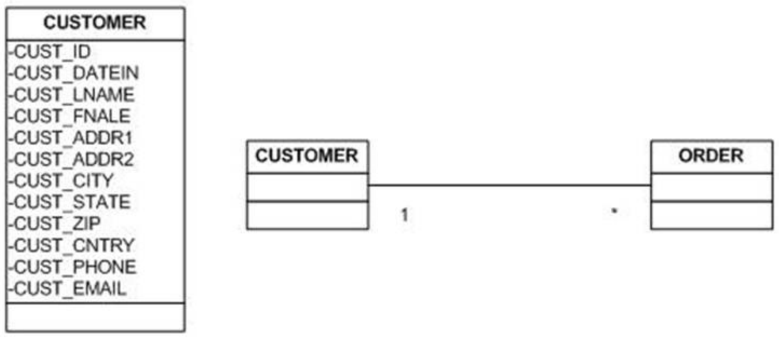


Figure 3.1 An example of an EIP and RIP, respectively

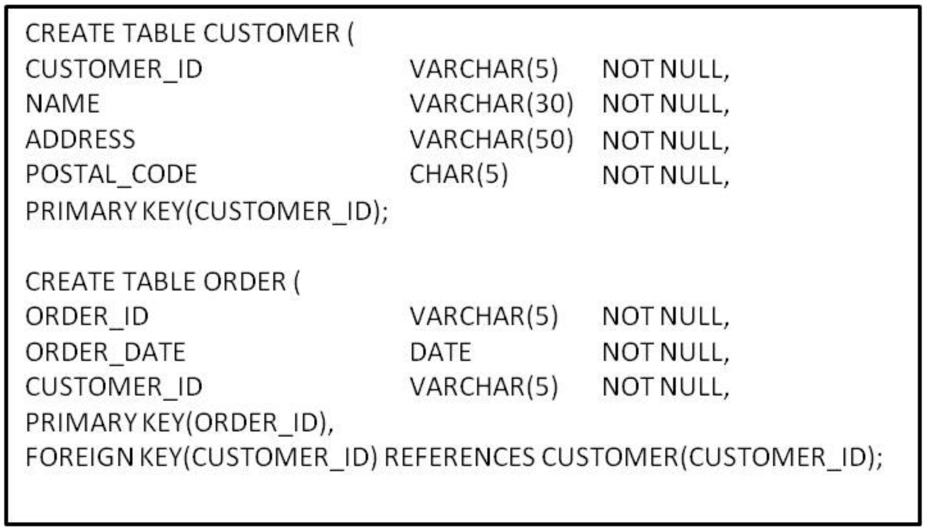
The method for creating EIR and RIR consists of the following three main steps:

INPUT: DDL schemas

OUTPUT: EIR and RIR

**1) Obtaining information about the executable schemas (DDL schemas)**

In order to reverse engineer existing database schemas, the information about the executable schemas must be available. These existing schemas (DDL schemas) have to provide at least relation names, attribute names, and PKs as seen in Figure 3.2.



```
CREATE TABLE CUSTOMER (
CUSTOMER_ID                  VARCHAR(5)    NOT NULL,
NAME                         VARCHAR(30)   NOT NULL,
ADDRESS                      VARCHAR(50)   NOT NULL,
POSTAL_CODE                  CHAR(5)       NOT NULL,
PRIMARY KEY(CUSTOMER_ID);

CREATE TABLE ORDER (
ORDER_ID                     VARCHAR(5)    NOT NULL,
ORDER_DATE                   DATE          NOT NULL,
CUSTOMER_ID                  VARCHAR(5)    NOT NULL,
PRIMARY KEY(ORDER_ID),
FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMER(CUSTOMER_ID);
```

Figure 3.2 The executable DDL schemas

In our research, we use a library of logical data models (executable schemas or DDL schemas) created by Silverston (Silverston, 2001) originally containing 464 relations and 1859 attributes as our first input. Later, the lists of EIR and RIR are extended by case studies.

**(2) Extracting EIP's elements**

We extract the EIP's elements from input DDL schemas by storing a relation name as an entity_name and an attribute as an attribute_name in EIR. The metadata model of EIP and RIP is shown in Figure 3.3.

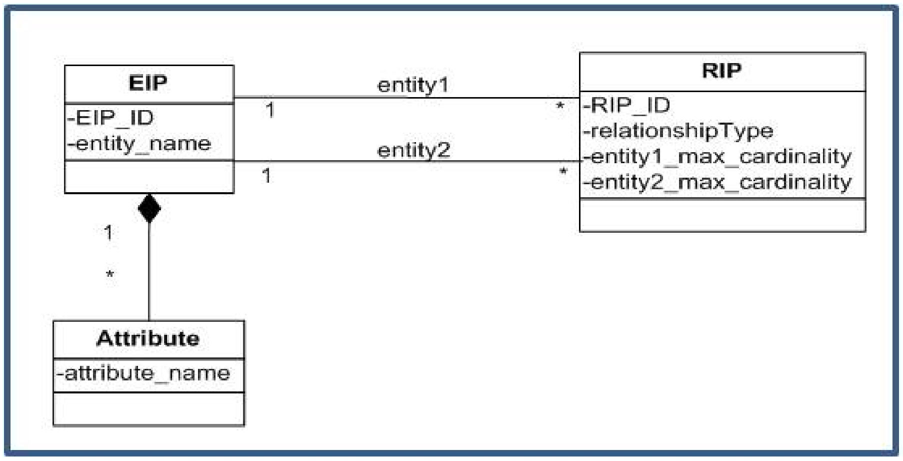


Figure 3.3 The metadata model of an EIP and an RIP

**(3) Extracting RIP's elements**

We extract the RIR's elements by identifying relationships between extracted entities obtained from Step (2) above. Most of the ER (Entity-Relationship) methods used in textbooks or CASE tools can be classified as either binary models or n-ary models (I.-Y. Song, Evans, M., Park, E. , 1995). There is an argument about the inclusion of binary or n-ary relationships in conceptual models (Hitchman, 2003). A central argument comes from the ability of n-ary modeling to reflect the true semantics of any given situation, whereas a binary model provides the simplest structures for a requirement specification's logical design. In database design, the binary relationship model is equivalently represented in a relational database management system. Many researchers suggest that all of the relationship representations in the conceptual model should be binary. For example, Rambaugh et al. (Rumbaugh, 1991) suggest that higher order relationships are more complicated to draw, implement, and understand than binary relationships, thus should be avoided if possible. In most cases, binary relationships are sufficient enough to represent the problem domain. Comprehensive analysis of binary relationships and ternary relationship is shown in (I.-Y. Song, Jones, T. , 1993). Therefore, in this research, we only specify the maximum cardinality constraints for binary models. Because of the limited semantic expressiveness of DDL schemas, the minimum cardinality cannot be automatically identified. Using a fully automated process, we can identify five relationship types:

*3.1 1: N for relationships identified by FK*

*3.2 1: N for relationships identified by partial keys*

*3.3 N: M for relationships identified by relationship relations*

*3.4 Is-a relationships*

*3.5 Recursive relationships*

Subsequently, these binary relationships are stored in RIR. The reverse engineering rules used in this step are created by inverting the schema transformation rules based on the EER (Extended Entity-Relationship) (Elmasri, 2004). These transformation rules are described as following:

### *3.1 1: N for relationships identified by FK*

**IF**: the PK of a relation $T_1$ is shown as a FK of another relation $T_2$,

**THEN**: there is a 1: N relationship between $T_1$ and $T_2$.

Consider these two relations:

$T_1$ ($\underline{K_1}$, $a_{11}$, $a_{12}$, $a_{13}$, …, $a_{1i}$)

$T_2$ ($\underline{K_2}$, $a_{21}$, $a_{22}$, $a_{23}$, …, $a_{2i,}$ $K_1^*$)

where $T_i$ represents a relation, $a_{ij}$ represents an attribute in a relation, PK is underlined, and FK is followed by a star symbol. If $T_2.K_1^*$ is a FK that comes from $T_1$, then there is a 1: N relationship between $T_1$ and $T_2$.

Ex. Consider these entities, EMPLOYEE and DEPARTMENT.

DEPARTMENT (DNAME<u>, DNUMBER</u>, MGRSSN*, MGRSTARTDATE)

EMPLOYEE (FNAME, LNAME, <u>SSN</u>, ADDRESS, SEX, SALARY, SUPERSSN, DNO*)

The PK, DNUMBER, of an entity DNUMBER, appears as a FK of entity EMPLOYEE. Then there is a 1:N binary relationship between entity DEPARTMENT and entity EMPLOYEE.

### *3.2 1: N for relationships identified by partial keys*

**IF:** the PK of a relation T1appears as a composite PK of another relation T2 and the PK of relation T1 is the FK of table T2 as well,

**THEN:** $T_1$ is a strong entity.

$T_2$ is a weak entity.

And there is a 1: N relationship between $T_1$ and $T_2$.

Consider these two relations:

$T_1$ ($\underline{K_1}$, $a_{11}$, $a_{12}$, $a_{13}$, …, $a_{1i}$)

$T_2$ ($\underline{K_1^{*}\ K_2}$, $a_{21}$, $a_{22}$, $a_{23}$, …, $a_{2i}$)

$T_2$ has a composite PK of ($K_1$, $K_2$) and only $K_1$ is a FK of table $T_2$, and $K_1$ is a PK of $T_1$. So, $T_1$ is a strong entity, $T_2$ is a weak entity, and there is a 1: N relationship between $T_1$ and $T_2$.

Ex. Consider these entities, EMPLOYEE and DEPENDENT.

EMPLOYEE (FNAME, LNAME, SSN, ADDRESS, SEX, SALARY, SUPERSSN, DNO*)

DEPENDENT (ESSN*, DEPENDENT_NAME, SEX, BDATE, RELATIONSHIP)

In this case, entity relations DEPENDENT has a composite PK of (ESSN, DEPENDENT_NAME), and only ESSN is a FK. Therefore, there is a 1: N relationship between entity EMPLOYEE and entity DEPENDENT.

***3.3 N: M for relationships identified by relationship relations***

Consider these two relations:

$T_1$ ($\underline{K_1}$, $a_{11}$, $a_{12}$, $a_{13}$, …, $a_{1i}$)

$T_2$ ($\underline{K_2}$, $a_{21}$, $a_{22}$, $a_{23}$, …, $a_{2i}$)

$T_3$ ($\underline{K_2^{*},\ K_1^{*}}$, $a_k$)

**IF**: $T_3$ has a composite primary key of ($K_2$, $K_1$), when consisting of FKs from the other two different tables $T_1$, and $T_2$,

**THEN**: there is a M : N relationship between $T_1$ and $T_2$.

Ex. EMPLOYEE (FNAME, LNAME, SSN, ADDRESS, SALARY, SUPERSSN, DNO*)

WORKS_ON (SSN*, PNO*, HOURS)

PROJECT (PNAME, PNO, PLOCATION, DNUM*)

Entity WORK_ON has a composite primary key of (ESSN, PNO), when consists of FKs from the entity EMPLOYEE and entity DEPARTMENT. So, there is an M: N relationship between entity EMPLOYEE and entity PROJECT.

### *3.4 Is-a Relationship*

**IF:** two strong entities, $T_1$ and $T_2$, have the same PK and $T_2$ has a key being both PK and FK,

**THEN:** $T_2$ has "Is-a" relationship with $T_1$ ($T_2$ Is-a $T_1$).

Consider these two relations:

$T_1$ ($\underline{K_1}$, $a_{11}$, $a_{12}$, $a_{13}$, …, $a_{1i}$)

$T_2$ ($\underline{K_1}^*$, $a_{21}$, $a_{22}$, $a_{23}$, …, $a_{2i}$)

Ex. EMPLOYEE(FNAME, LNAME, SSN, ADDRESS, SALARY, SUPERSSN, DNO*)

MANAGER (SSN*, RANK, PROMOTION_DATE, DEPTNO)

In this case, the relations, EMPLOYEE and MANAGER, have the same PK (SSN), and MANAGER has SSN as being both PK and FK. This suggests that there is an "Is-a" relationship exists from relation MANAGER and relation EMPLOYEE.

### *3.5 Recursive Relationship*

**IF**: $T_1$ has a FK that references the PK of its own table ($T_1$),

**THEN**: $T_1$ has recursive relationship.

Consider this relation:

$T_1(\underline{K_1}, a_{11}, a_{12*}, a_{13*}, \ldots, a_{1i})$

Ex. EMPLOYEE (FNAME, LNAME, SSN, ADDRESS, SEX, SALARY, SUPERSSN*, DNO*)

In this case, each Employee occurrence contains two social security numbers (SSN), one identify the employee, the other being the SSN of the employee's supervisor.

## 3.2 The Six Domain Independent Modeling Rules

This section presents our selected six modeling rules termed as the six domain independent modeling rules. Our survey shows that one of the difficulties in creating conceptual models is the scattered modeling rules. There is no complete set of rules that help developing conceptual models. In general, rules/heuristics are useful but sometimes they may lead to cognitive errors called bias (D. Batra, 2007; Parson, 2004). There is always trade off in design so that not all rules can work together because some rules are conflicting. We have selected the six domain independent modeling rules based on teaching experiences of over 20 years by one of the committee members of this dissertation. These six rules are considered as a minimal set of rules to teach novice designers in creating quality conceptual models. These six rules are not based on the syntax of any NLs and thus are domain independent. This means that these rules can be applied to a wide range of applications and domains. In this research, we would like to experiment whether the six rules are indeed useful. The six domain independent modeling rules are:

**R1: The ID (Identifier) Rule**

IF a concept (noun or verb) needs to have a unique identifier, THEN it can be an entity.

**R2: The MA (Multiple Attribute) Rule**

IF a concept has multiple attributes, THEN it can be an entity.

**R3: The MVA (Multi-Valued Attribute) Rule**

IF a concept has multi-values, THEN it can be an entity.

**R4: The TDA (Time-dependent attributes) Rule**

IF a concept has time-dependent attributes or needs to keep track of history of values, THEN it can be an entity.

**R5: The SC (Single Concept) Rule**

A good entity should represent one and only one concept.

**R6: The DI (Domain Importance) Rule**

IF a concept is important in its own right within the problem domain whether it has one or multiple attributes, THEN it can be an entity.

## 3.3 Developing EIPW (Entity Instance Pattern WordNet)

In this research, we explore knowledge-based and pattern-based approaches that help database designers develop quality conceptual data models. We propose new types of reusable pattern artifacts, called the entity instance repository (EIR) and the relationship instance repository (RIR), which are repositories of entity instance patterns (EIPs) and relationship instance patterns (RIPs), respectively.  In the previous section, EIR and RIR have been created as the reusable pattern repositories containing knowledge about an application domain. This section discusses how these patterns are implemented in KBS called EIPW (Entity Instance Pattern WordNet).

### 3.3.1 Overview of EIPW Architecture

The architecture of EIPW is shown in Figure 3.4. A prototype of EIPW has been developed by using JAVA Applet. Firstly, the system passes a NL requirement specification as an input to do the part of speech tagging (POS) in order to list all of the possible candidate entities. We use a well-known open source called LingPipe (http://alias-i.com/lingpipe) to perform POS. In EIPW, the entity list can be identified based on noun phrases and hidden requirements. During the post-parsing analysis, a noun phrase belonging to any of *a discard noun set* will be excluded as a candidate entity. The discard noun set are created based on the history of words discarded by designers and the class elimination rules (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). The discard noun set is domain independent.

In the entity identification module, there are three activities performed:

1. The first activity is to identify the entity list based on EIR. WordNet is also used to ensure that the synonyms of EIR's entities are not missed out while preparing the lists of entities.

2. The second activity is to identify the entities that are not detected by EIR by applying the top noun categories and hypernym chains in WordNet (Du, 2008).

3. The third activity is to identify the hidden entities by applying entity categories. Our entity categories are adopted from the class categories defined by Song et al. (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004).

Relationships between entity lists are generated by considering the application domain semantics inherent in the RIR. The modeling rules are used to ensure that all of the relationships are identified. The lists of EIR and RIR are extended by case studies. WordNet is also used to ensure that the synonyms of EIR's entities and RIR's entities are missed out while preparing the list of candidate entities and preparing the list of relationships, respectively.

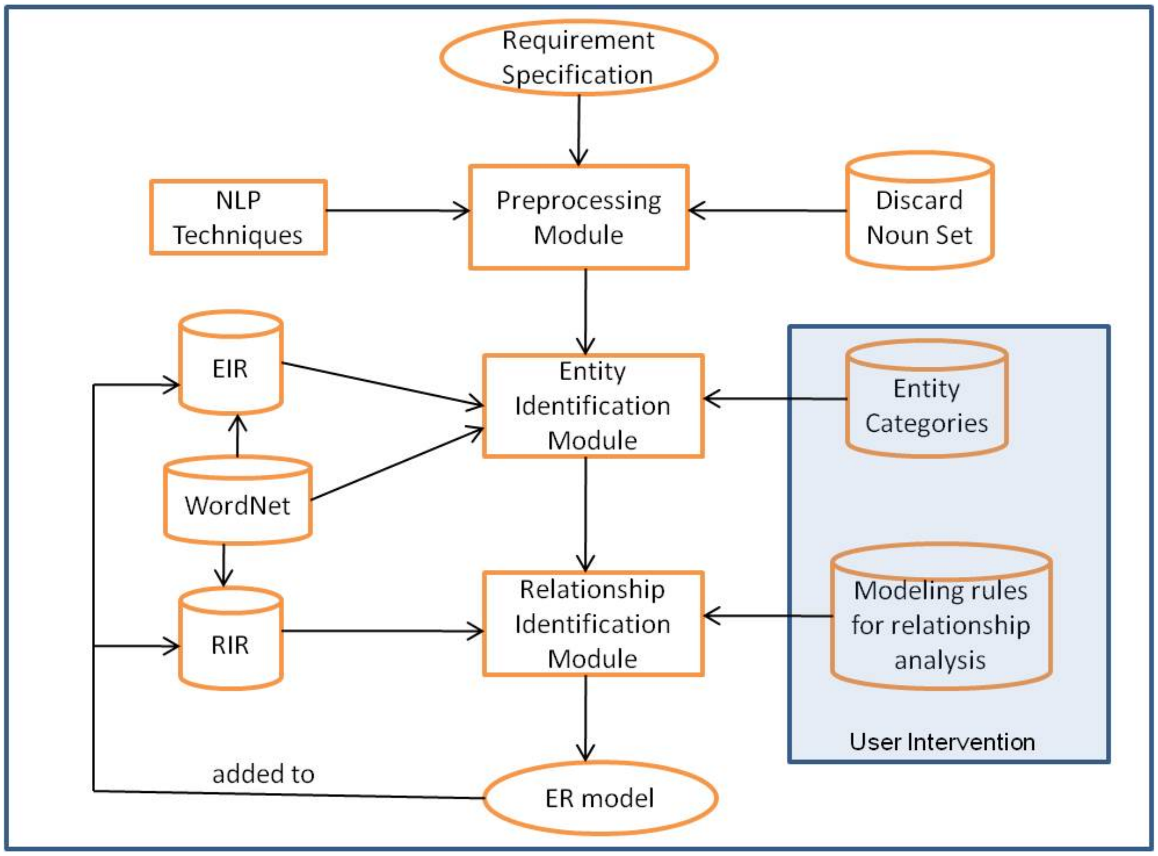

Requirement Specification
NLP Techniques
Preprocessing Module
Discard Noun Set
EIR
Entity Identification Module
Entity Categories
WordNet
RIR
Relationship Identification Module
Modeling rules for relationship analysis
User Intervention
added to
ER model

Figure 3.4 The EIPW architecture

## 3.3.2 The EIPW Workflow

This section shows the detailed workflow of EIPW and its use for generating ER models. EIPW can be mainly divided into two subtasks: entity identification and relationship identification.

## (1) Entity Identification

The actual step-by-step activities of our methodology outlined in Figure 3.6 are the form of an activity diagram in the UML. In Figure 3.6, the three swimlanes perform the following activities:

- The middle swimlane: The aim of these swimlane activities is to identify entities based on EIR.
- The rightmost swimlane: The aim of these swimlane activities is to identify entities that are not detected by EIR by applying the top noun categories and hypernym chains in WordNet.
- The leftmost swimlane: The aim of these swimlane activities is to identify hidden entities that are not explicitly stated in the requirements but are necessary for the conceptual modeling by applying entity categories. Entities categories are used as a tip for identifying entities.

The details of the activities in Figure 3.6 are presented below.

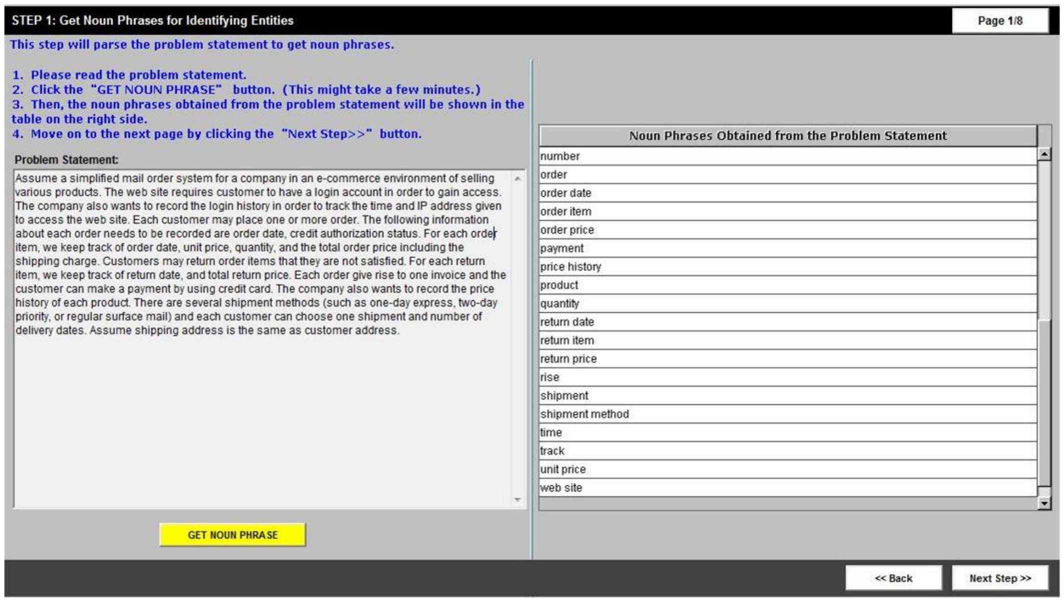
**Activities of Middle Swimlane of Figure 3.6**

- Begin with a requirement and remove the partial explanation statements

  This process starts by reading a text file containing a requirements specification of an application written in English. Explanation statements in a requirements specification aim to help human readers to understand the requirements better but they are harmful for automated requirement analysis (Du, 2008). For example, in "A new video store intends to offer rentals (and sales) of entertainment material to the wider public." the explanation part inside the parenthesis is redundant and not necessary to transform it into ERD representation. Heuristic rules based on parenthesis and some words (e.g. such as) will be used to remove the explanation statement.

- *Step 1: Get noun phrases*

  Part of speech tagging (POS) is used to assign each word in an input sentence its proper part of speech such as noun phrases and verb phrases in order to reflect the word's syntactic categories. The POS tags provide a useful abstraction of words whereby candidate entities can be identified from either noun phrases (P. Chen, 1983) or verb phrases (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). In this research, we use an open source call LingPipe for POS tagging to get the entire noun phrases appearing in the requirement specification. Figure 3.5 shows the user interface of Step 1 that lists all the noun phrases appearing in the requirement specification.



STEP 1: Get Noun Phrases for Identifying Entities

Page 1/8

This step will parse the problem statement to get noun phrases.

1. Please read the problem statement.
2. Click the "GET NOUN PHRASE" button. (This might take a few minutes.)
3. Then, the noun phrases obtained from the problem statement will be shown in the table on the right side.
4. Move on to the next page by clicking the "Next Step>>" button.

Problem Statement:

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

| Noun Phrases Obtained from the Problem Statement |
|---|
| number |
| order |
| order date |
| order item |
| order price |
| payment |
| price history |
| product |
| quantity |
| return date |
| return item |
| return price |
| rise |
| shipment |
| shipment method |
| time |
| track |
| unit price |
| web site |

Figure 3.5 The EIPW's user interface of Step 1

- *Step 2: Test discard noun set*

  To facilitate the post-parsing analysis, a noun phrase belonging to any of the discard noun set will be excluded as a candidate entity. The discard noun set is created based on the history of words discarded by designers and the class elimination (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004) to provide intelligent suggestion for better parsing the requirement. The discard noun set is domain independent. Some examples of discard noun set are number, ID, information, database, track, record, system, etc.

- *Step 3: Identify entities based on EIR*

After the initial set of possible candidate entities is identified from Step 2, each term is compared to the entity names in the EIR. If a noun phrase matches an entity name in EIR, then it becomes a candidate entity (E1).

- *Step 4: Apply WordNet Synonym*

  Out of these entity names, identify synonyms of a noun phrase from WordNet. If the synonyms of a noun phrase match the entity name in EIR, then the noun phrase also becomes a candidate entity (E1).

**Activities of RightMost Swimlane of Figure 3.6**

- *Step 1: Apply top noun categories and hypernym chains in WordNet*

  The top noun categories and hypernym chains in WordNet are used to perform entity categorizations. The entity categories can help us identify entities from non-identifiable noun phrases from the MiddleMost Swimlane. These entity categories can be divided into two groups and defined as follows:

  - Potential-Entity Group: group, physical object, physical entity, thing, transaction.
  - Non-Entity Group: cognition, attribute, value, measure, constituent, language_unit, feeling.

  If the hypernym chain of a noun phrase reaches to one of the categories in the "Potential-Entity" group, the system will label this term as a candidate entity (E2) and insert it to the EIR to expand the list of EIP. On the other hand, if the hypernym chain of a term reaches to one of the categories in the “Non-Entity” group, the system will delete this noun phrase.

If the hypernym chain of a term does not belong to either group, the system will ask the user to make judgments regarding this term. If the user labels it as a candidate entity (E2), then insert it to the EIR, else delete this noun phrase.

**Activities of LeftMost Swimlane of Figure 3.6**

In this swinlane, the system asks the user to identify the hidden entities by applying domain knowledge to entity categories. Our entity categories in business applications adopt the class categories defined by Song et al. (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). They are as follows:

1. *Roles of People*
   They represent role of humans who perform some important function.
   Ex. Student, Employee, Customer.
2. *Places*
   They represent locations where important business activities are occurred.
   Ex. Warehouse, Brach, Store.
3. *Physical Things*
   They represent tangible objects that are import in business activities.
   Ex. Product, Machine, Device, Book
4. *Organization*
   They represent important business units.
   Ex. Department, Team
5. *Events (Transactions)*

They represent important activities that need to record some data with the time the event occurred.

Ex. Order, Promotion, Payment

6. *Transaction Line Items*

   They represent an element of a transaction.

   Ex. Order-Line-Item, Purchase-Line-Item, Rental-Line-Item

7. *Concepts with properties*

   They represent intangible ideas used to keep track of business activities.

   Ex. Project, Account, Complaint

8. *Specification*

   They represent a description of other items that need to be distinguished from one another.

   Ex. Video-Title, Flight-Plan. For example, each video tape has a different barcode.

9. *Interaction*

   They represent an association between two entities, where the association has meaningful attributes. An example of this entity is Reservation between Passenger and Flight entities.

10. *Rules/Policies/Reference/Look up*

    They represent important business rules.

    Ex. Rental-Policy, Shipping Method

11. *Containers of other things*

    They represent entities that will contain other entities.

    Ex. Shelf, Catalog, Pick List, Bin

12. *Things in a container*

    They represent entities that will be contained in another entity.

    Ex. Order-Line-Item, Passenger, Video-Title in a catalog.

- *Step 1: Apply domain knowledge to entity categories (user intervention)*

  For each entity category, check whether all the entities representing the entity category are already captured. Otherwise create a new candidate entity (E3) based on the entity categories.

A set of entities identified from our methodology is a union of the entities identified from the three swimlanes. That is: $\{\text{Entities}\} = \{E1\} \cup \{E2\} \cup \{E3\}$

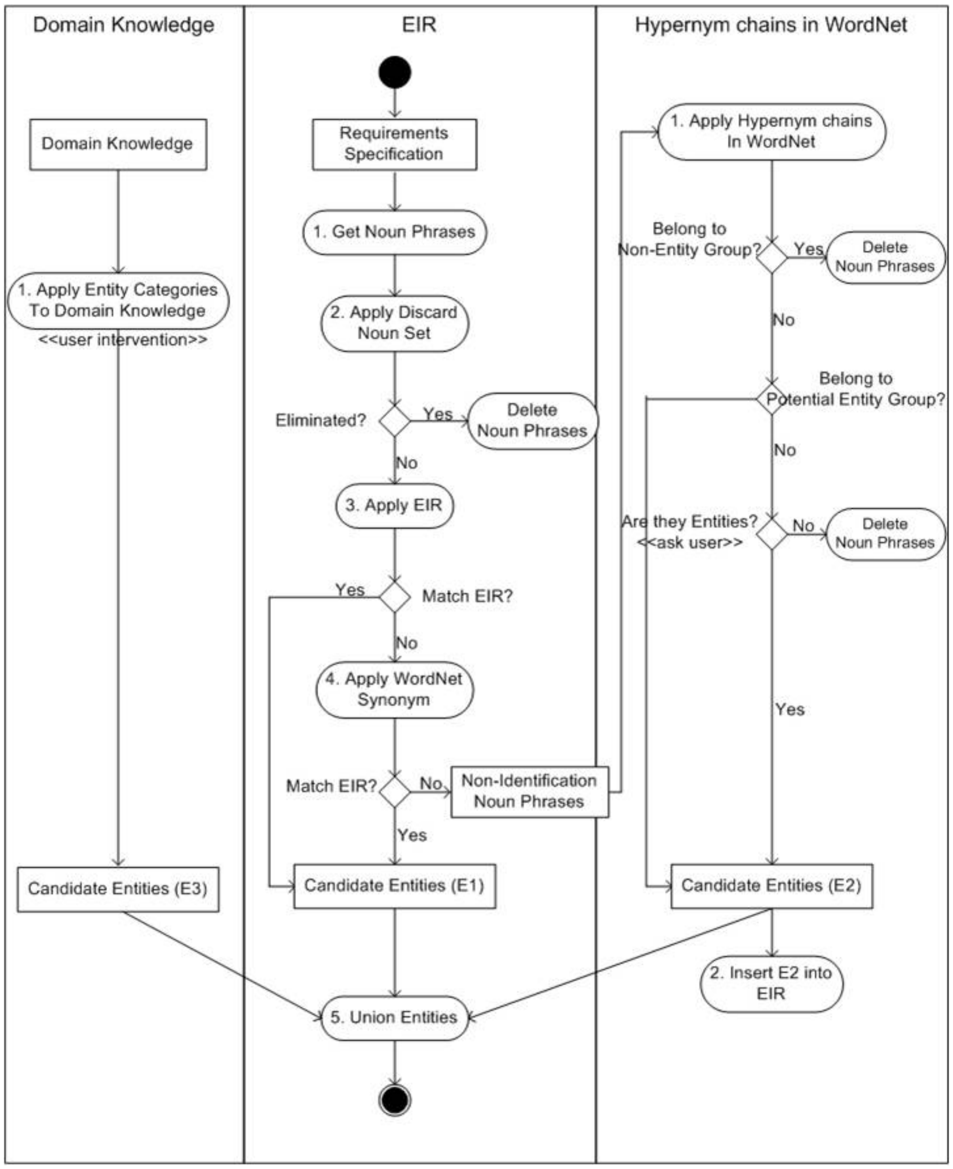


Figure 3.6 Entity Identification Process in EIPW

## (2) Relationship Identification

After the entity list has been identified in the entity identification process, Relationships between entities are generated by considering the application domain semantics inherent in the RIR. This repository is used to identify occurring relationships within an application domain and to generate the relationships between entities. The flowcharts for the relationship identification process are shown in Figure 3.7. The processing task requires several activities in order to determine the relationships between the entities. In figure 3.7, there are two swimlanes performing the following activities.

- **The left swimlane:** The goal of these swimlane activities is to identify relationships (r) between candidate entities based on RIR.
- **The right swimlane:** The goal of these swimlane activities is to ask the users to identify the relationships, which are not detected by the RIR, by applying Need-to-Remember Rule (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004).

The details of the activities in Figure 3.6 are discussed below.

**Activities of left swimlane of Figure 3.7**

- Begin with the candidate entity list obtained from entity identification process.
- Step 1: Delete duplicate entities.

  This Step is conducted through WordNet synonyms.
- Step 2: Assign all possible relationships ($r_{ij}$) between the candidate entities.
- Step 3: Match the possible relationships ($r_{ij}$) with RIR.

  If $r_{ij}$ match the relationships in RIR, add $r_{ij}$ into Relationship set (R).
- Step 4: Apply WordNet Synonym

  Out of the matching, identify synonyms of entity names from WordNet. If the synonyms of $r_{ij}$ match the relationship in RIR, add $r_{ij}$ in R.

**Activities of right swimlane of Figure 3.7**

- Begin with the possible relationships that are not detected by RIR from left swimlane.
- *Step 5: Apply Need-to-Know rule (user intervention)*

  If a relationship represents an association that does not have to be remembered between two entities, then delete this relationship.
- *Step 6: Assign the multiplicity (user intervention)*

  Assign the multiplicity to each relationship obtained from Step 5.

The ER model is created by combining a set of relationships (R) identified from the two swimlanes. Figure 3.8 shows the output of EIPW.

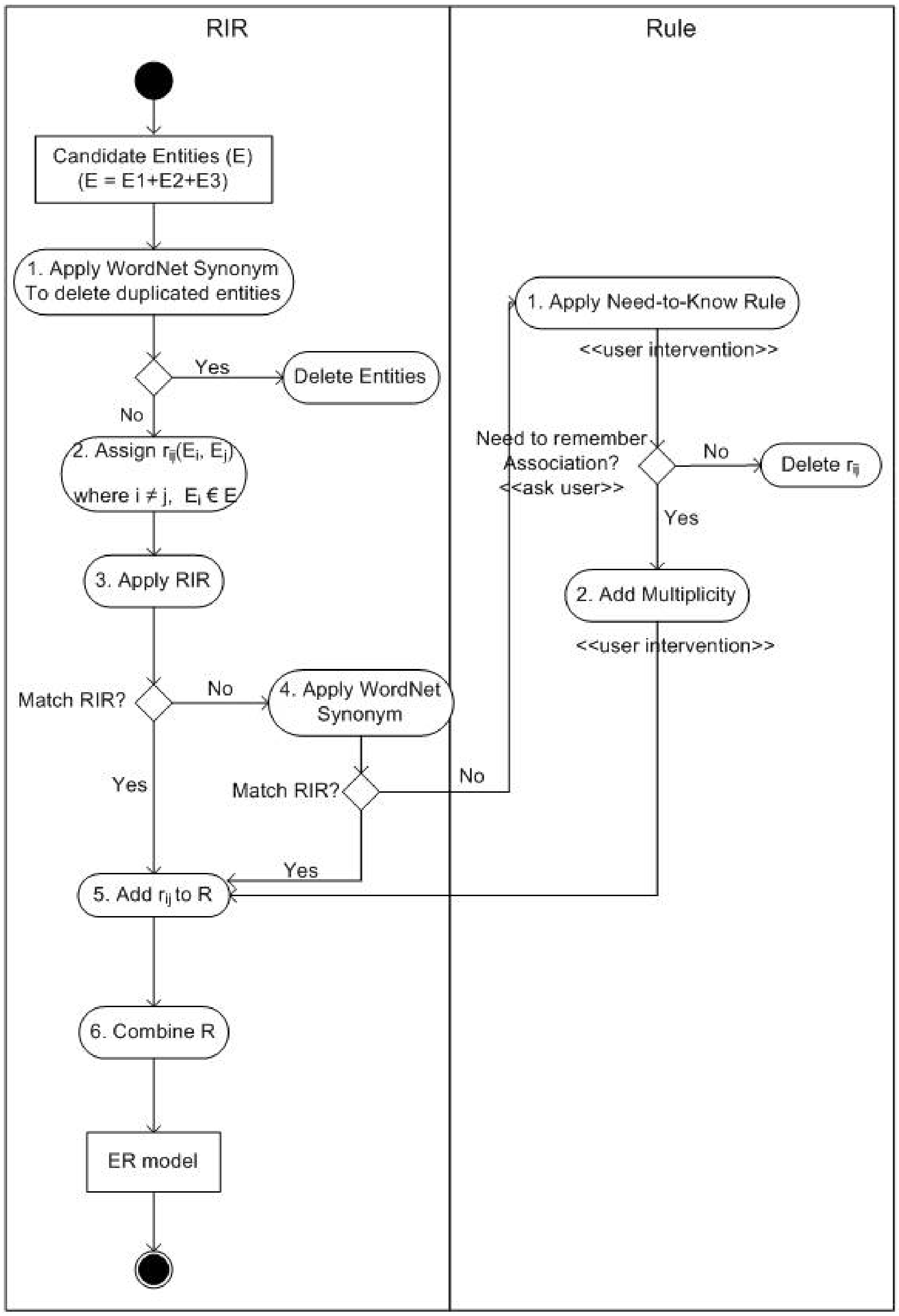


Figure 3.7 Relationship Identification Process in EIPW

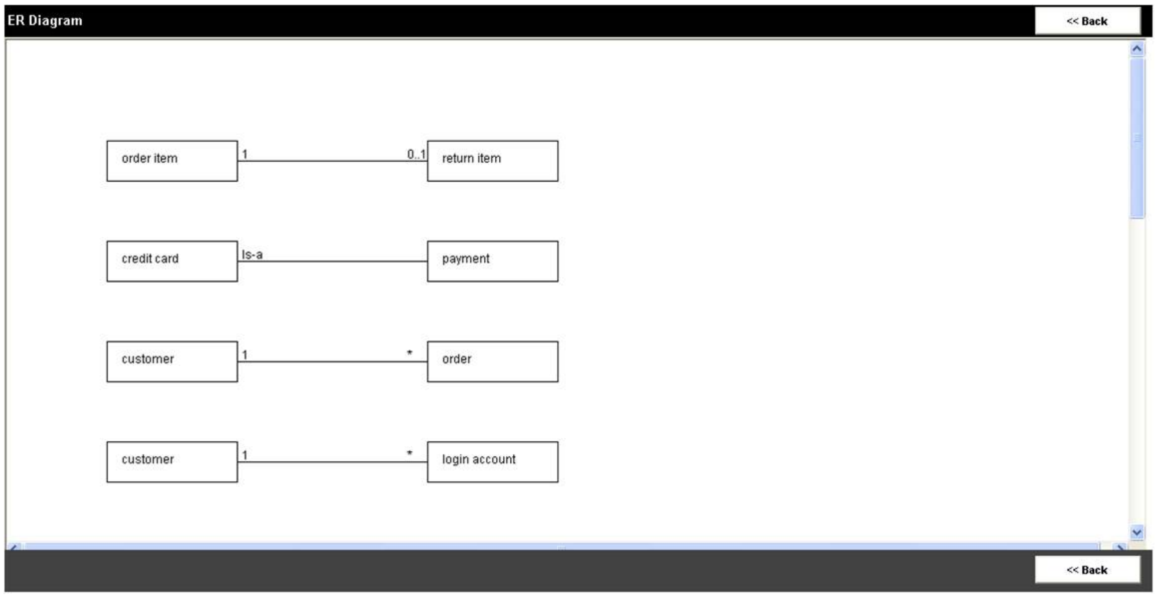


Figure 3.8 The output of EIPW

## 3.4 Developing HBT (Heuristic-based Technique)

In this research, we select six domain independent modeling rules that are comprehensive enough in creating conceptual models. We evaluate the usefulness of these rules by developing HBT that applies these rules to the creation of conceptual data models. The knowledge implemented in the system is based on the six domain independent modeling rule. The six domain rules are used to ensure whether the initial candidate entity lists should be included or excluded in the data model.

### 3.4.1 Overview of HBT Architecture

The system modules are shown in Figure 3.9. A prototype of HBT was developed by JAVA applet. First, the system takes a NL requirement specification as an input to a preprocessing module. The main functionality of the preprocessing module is to do the POS in order to list all of the possible candidate entities. In HBT, the entity list can be identified based on noun phrases, verb phrases, and hidden requirements. During the post-parsing analysis, a noun phrase and a verb phrase belonging to any of a discard noun set and a discard verb set, respectively, will be excluded as a candidate entity. The discard noun set and the discard verb set are created based on the history of words discarded by designers and the class elimination rules (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004). The discard noun set and the discard verb set are domain independent. Most of the other modules' functions in HBT are very similar to those in EIPW. Also, the user interfaces of HBT are also similar to those in the HBT as shown in Figure 3.10. The only difference is in the entity identification module. In this module, there are three activities performed:

1. The first activity is to identify the entity list based on noun phrases by using the six domain independent modeling rules, which are the ID, MA, MVA, TDA, SC, and DI rules.

2. The second activity is to identify the entity list based on verb phrases by using two rules out of six domain independent modeling rules, which are the ID and MA rules.

3. The third activity is to identify hidden entities that are not explicitly stated in the requirements but are necessary for the conceptual modeling by applying entity categories.

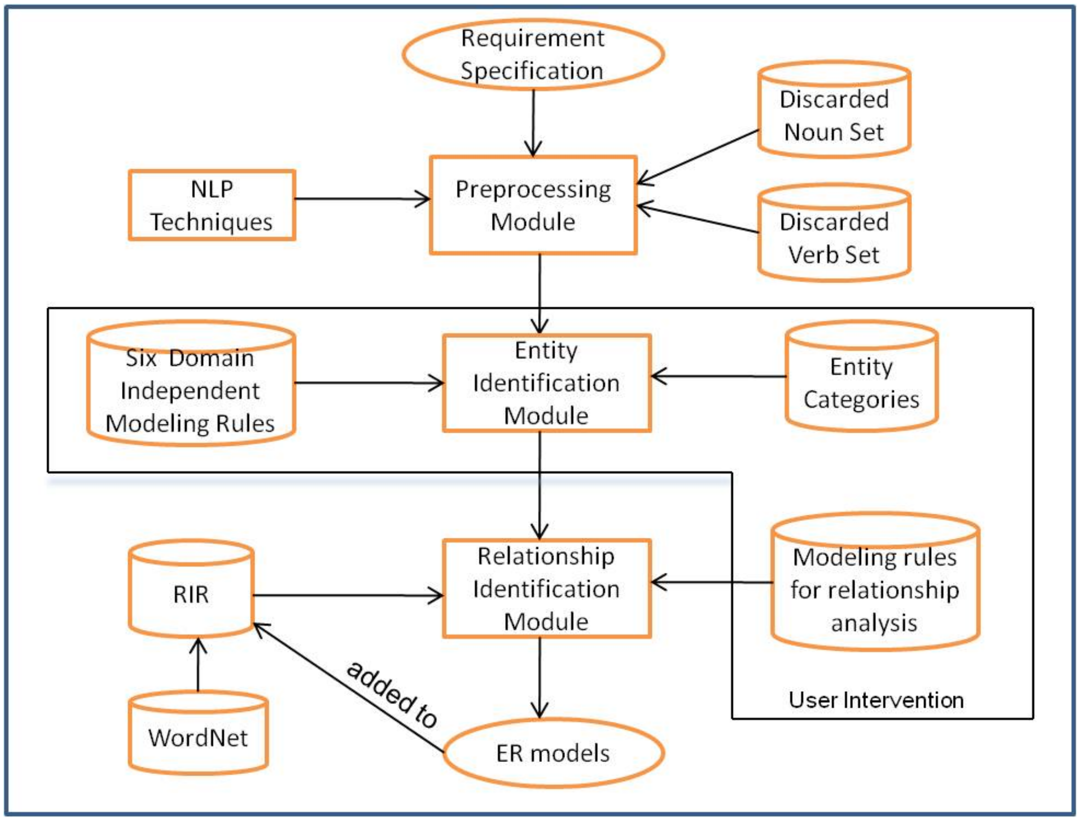
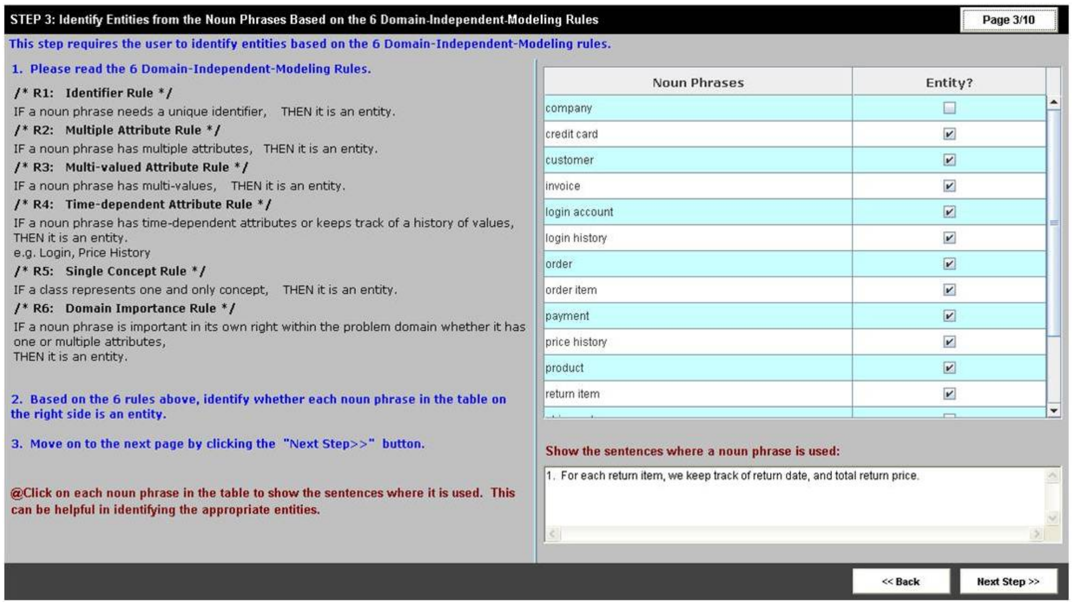

Requirement Specification
Discarded Noun Set
NLP Techniques
Preprocessing Module
Discarded Verb Set
Six Domain Independent Modeling Rules
Entity Identification Module
Entity Categories
RIR
Relationship Identification Module
Modeling rules for relationship analysis
added to
User Intervention
WordNet
ER models

Figure 3.9 The HBT Architecture



STEP 3: Identify Entities from the Noun Phrases Based on the 6 Domain-Independent-Modeling Rules

Page 3/10

This step requires the user to identify entities based on the 6 Domain-Independent-Modeling rules.

1. Please read the 6 Domain-Independent-Modeling Rules.

**/* R1: Identifier Rule */**
IF a noun phrase needs a unique identifier, THEN it is an entity.
**/* R2: Multiple Attribute Rule */**
IF a noun phrase has multiple attributes, THEN it is an entity.
**/* R3: Multi-valued Attribute Rule */**
IF a noun phrase has multi-values, THEN it is an entity.
**/* R4: Time-dependent Attribute Rule */**
IF a noun phrase has time-dependent attributes or keeps track of a history of values, THEN it is an entity.
e.g. Login, Price History
**/* R5: Single Concept Rule */**
IF a class represents one and only concept, THEN it is an entity.
**/* R6: Domain Importance Rule */**
IF a noun phrase is important in its own right within the problem domain whether it has one or multiple attributes,
THEN it is an entity.

2. Based on the 6 rules above, identify whether each noun phrase in the table on the right side is an entity.

3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase in the table to show the sentences where it is used. This can be helpful in identifying the appropriate entities.

| Noun Phrases | Entity? |
|---|---|
| company | ☐ |
| credit card | ✓ |
| customer | ✓ |
| invoice | ✓ |
| login account | ✓ |
| login history | ✓ |
| order | ✓ |
| order item | ✓ |
| payment | ✓ |
| price history | ✓ |
| product | ✓ |
| return item | ✓ |

Show the sentences where a noun phrase is used:

1. For each return item, we keep track of return date, and total return price.

Figure 3.10 An example of user interface in HBT

### 3.4.2 The HBT Workflow

This section presents the detailed workflow of HBT. HBT incorporates the six domain independent modeling rules, entity categories, and relationship instance repository. HBT's process can be divided into two subtasks: entity identification and relationship identification. In entity identification process, the six domain independent modeling rules are used to ensure whether the initial candidate entity lists should be included or excluded in the data model. The relationship identification process of HBT is the same as that used in EIPW.

### 1) Entity Identification

The step-by-step activities of our methodology outlined in Figure 3.11 are in the form of an activity diagram in the UML. The processing task requires several steps to be carried out in order to achieve the candidate entities from the NL input. In Figure 3.11, there are three swimlanes to perform the following activities:

- The middle swimlane: The aim of these swimlane activities is to identify entities based from the concepts that are explicitly stated as noun phrases in the requirements.
- The rightmost swimlane: The aim of these swimlane activities is to identify entities based from the concepts that are explicitly stated as verb phrases in the requirements.
- The leftmost swimlane: The aim of these swimlane activities is to identify hidden entities that are not explicitly stated in the requirements but are necessary for the conceptual modeling. We identify those hidden entities by applying domain knowledge to entity categories.

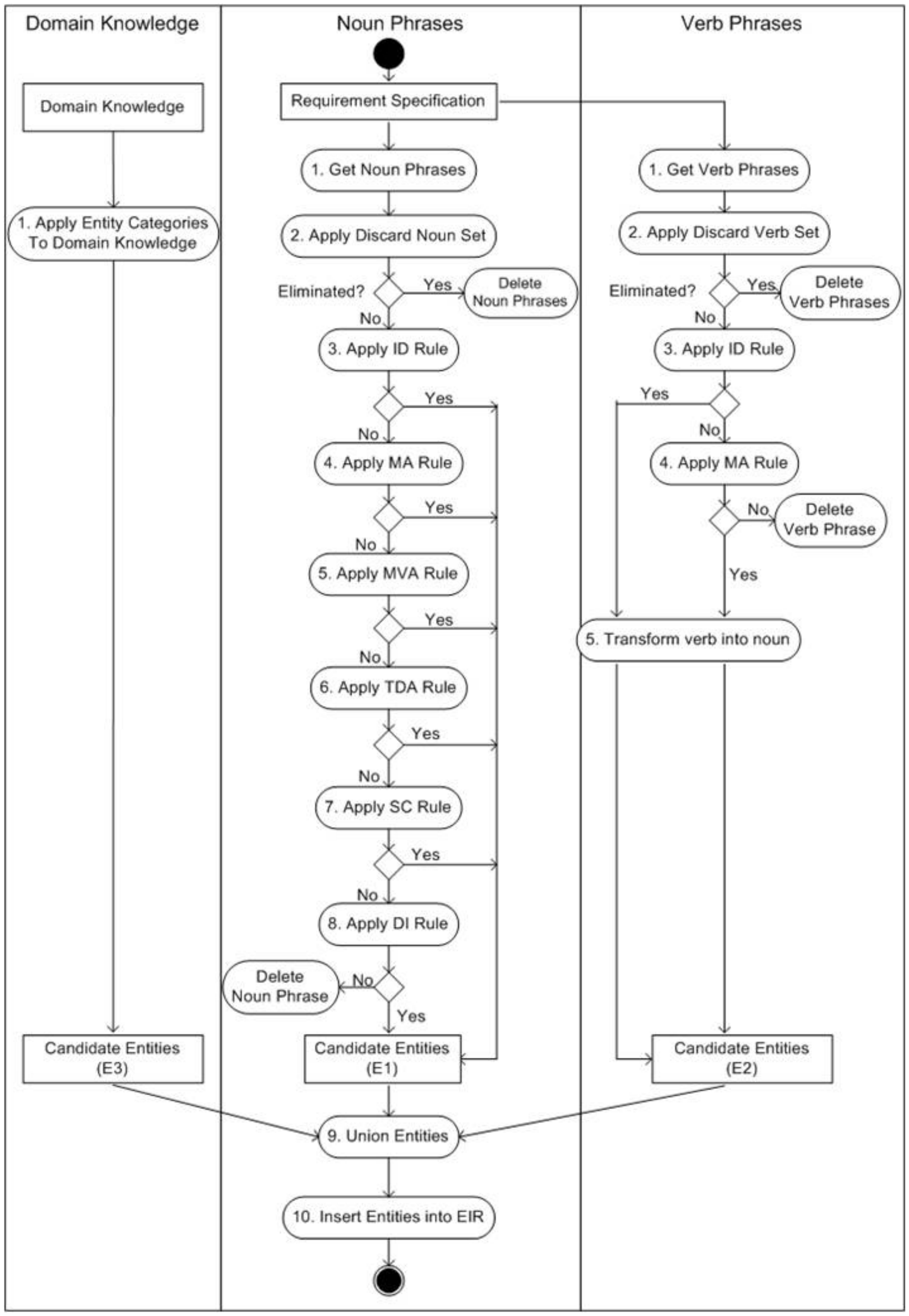


Figure 3.11 Entity Identification Process in HBT

**Activities of Middle Swimlane of Figure 3.11**

- Begin with a requirements specification and remove the partial explanation statement.
- *Step 1: Get noun phrases*

  Get the entire noun phrases from a requirement specification by using POS technique.
- *Step 2: Test discard noun set*

  To facilitate the post-parsing analysis, noun phrases belong to any of the discard noun set will be deleted.
- *Step 3:  Apply the ID (Identifier) Rule*

  If a noun phrase needs to have a unique identifier, then it is a candidate entity (E1).
- *Step 4: Apply the MA (Multiple Attributes) Rule*

  If a noun phrase has multiple attributes, then it is a candidate entity (E1).
- *Step 5: Apply the MVA (Multi-Value Attribute) Rule*

  If a noun phrase has multi-values, then it is a candidate entity (E1).
- *Step 6: Apply the TDA (Time-dependent attributes) Rule*

  If a noun phrase has time-dependent attributes or needs to keep track of history of values, then it is a candidate entity (E1).
- *Step 7: Apply the SC (Single Concept) Rule*

  A good entity should represent one and only one concept (E1).

- *Step 8: Apply the DI (Domain Importance) Rule*

  If a noun phrase is important in its own right within the problem domain whether it has one or multiple attributes, then it is a candidate entity (E1).

**Activities of Rightmost Swimlane of Figure 3.11**

- *Step 1: Get verb phrases*

  We use POS technique to get all the verb phrases from the requirement specification.

- *Step 2: Test the discard verb set*

  To facilitate the post-parsing analysis, verb phrases belong to any of the discard verb set will be deleted. The examples of verb phrases in discard verb set are: automate, become, concern, etc.

- *Step 3: Apply ID (Identifier) Rule*

  If the concept represented by a verb phrase needs to have a unique identifier, then it is a candidate entity (E2). Song et al. (I.-Y. Song, Yano. K., Trujillo, J., Lujan-Mora, S. , 2004) term this candidate entity a Transformed Entity.

- *Step 4: Apply MA (Multiple Attribute) Rule*

  If the concept represented by a verb phrase has multiple attributes, then it is a candidate entity (E2).

**Activities of Left Swimlane of Figure 3.11**

In this swimlane, the system asks the user to identify the hidden entities by applying domain knowledge to entity categories. These entity categories are the same as that used in the EIPW.

- *Step 1: Apply domain knowledge to entity categories*

  For each entity category, check whether all the entities representing the entity category are already captured. Otherwise create a new entity based on the entity category.

A set of entities (E) identified from our methodology is a union of the entities gotten from the three swimlanes. The relationship identification process of HBT is the same as that used in EIPW. The output of HBT is shown in Figure 3.12.

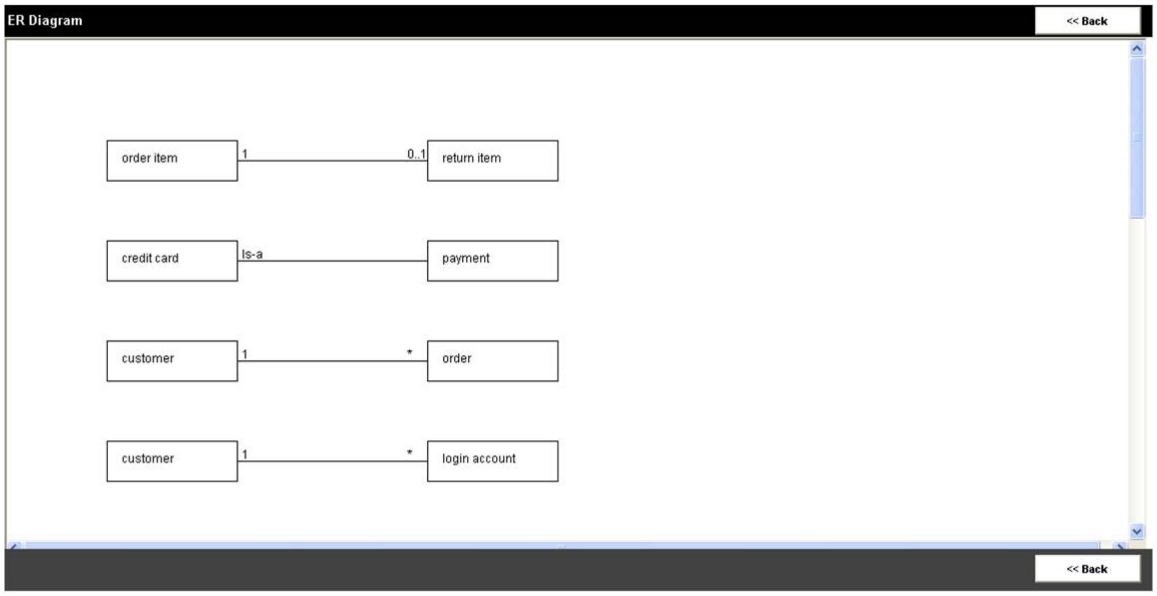


Figure 3.12 The output of HBT

# 4. EMPIRICAL EVALUATION

In this research, we have proposed methods for improving the process of conceptual modeling design. We have implemented two knowledge-based data modeling tools: EIPW and HBT. EIPW incorporates entity instance repository, entity categories, relationship instance repository, and WordNet. HBT incorporates the six domain independent rules, entity categories, and relationship instance repository.

In this Chapter, we evaluated the quality of outputs generated by EIPW and HBT by using ANOVA technique. Because the quality of the ER models is of interest, the following hypotheses are tested:

*H1: Novice designers using EIPW will create conceptual models with better quality compared to the models generated without using any tools.*

*H2: Novice designers using HBT will create conceptual models with higher scores compared to the models generated without using any tools.*

*H3: There is no significant difference between the two KBSs regarding the quality of the conceptual models.*

## 4.1 Experiment Design

The experimental framework is shown in Figure 4.1. The two independent variables are the systems and the task sizes. In conceptual modeling, a linear increase in the number of entities can result in a combinatorial increase in the number of possible relationships (D. Batra, 2007). As the task size increases, so do the numbers of decisions required in the modeling process. Therefore, our experiment design incorporates two levels of the task size to provide some sensitivity for this factor. The medium task size has 9 entities and 9 relationships, while the moderate task size has 14 entities and 14 relationships. The dependent variable is the quality scores of the ER models.

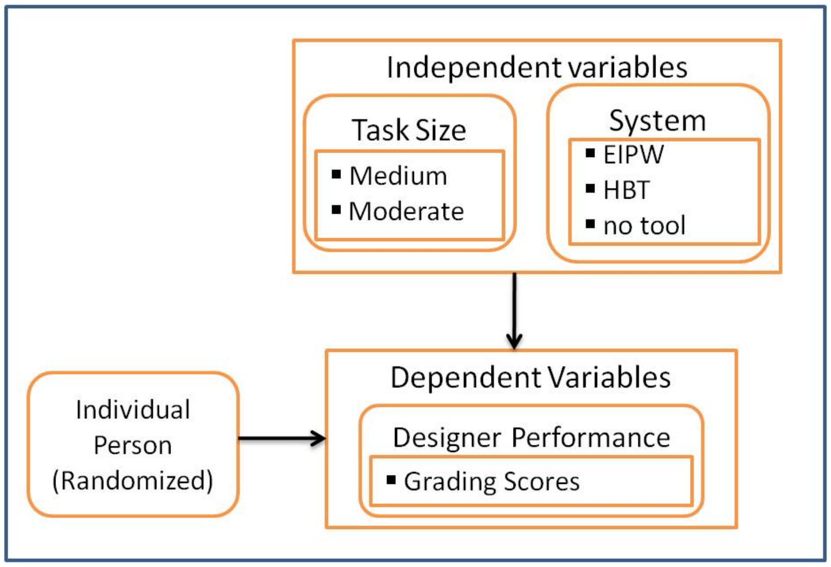


Figure 4.1 The framework of empirical experiments

## 4.2 Subjects and Tasks

There were 41 subjects. All of the subjects were students in the iSchool at Drexel University and did not work in conceptual modeling field before. Therefore, we concluded that all of our subjects were novice designers. Twenty-one were undergraduates and twenty were graduate students. Forty-one subjects were divided into four groups as shown in Table 4.1. Each subject worked on four problem statements: one medium size and one moderate size problem statements with the aid of our KBS, and one medium size and one moderate size problem statements with no tool. The problem statements are in the e-commerce domain. The subjects could take time as long as they wanted to create conceptual models based on the given problem statements.

Table 4.1 The Experiment Design

| **Group** | **Num of subject** | **Problem1** | **Problem2** | **Problem3** | **Problem4** |
|---|---|---|---|---|---|
| 1 | 11 | No tool | No tool | Using EIPW | Using EIPW |
| 2 | 10 | Using EIPW | Using EIPW | No tool | No tool |
| 3 | 10 | No tool | No tool | Using HBT | Using HBT |
| 4 | 10 | Using HBT | Using HBT | No tool | No tool |

## 4.3 Evaluation Metrics

1. Evaluating experimental data
   The quality of the conceptual data models created by the novice designers is judged by a third party.
2. Grading criteria
   The quality of an ER model is evaluated by a scoring schema that specifies how to grade the ER model on each facet (entities and relationships). In this research we adopt the scoring scheme proposed by Du (Du, 2008). It focuses on the correct identification of appropriate entities and relationships based on the given problem statements.

   Entity

   - Add 2 points for each correct entity stated in the problem statement.
   - No penalty for very likely entity but not stated in the problem statement.
   - Deduct 1 point for each wrong entity.
   - Deduct 1 point for each missing entity.

<u>Relationship</u>

- Add 2 points for each relationship that is correctly attached to the corresponding entities.
- Deduct 1 point for each missing relationship.
- Deduct 1 point for each a redundant relationship or a derivable relationship.
- Deduct 1 point for each wrong relationship.
    - Shown as an indirect relationship without a direct relationship.
- Deduct 1 point for each incorrect degree of relationship.
- Deduct 0.5 point for each an incorrect cardinality.

## 4.4 Empirical Results

### Test of Hypothesis 1: EIPW

A *2x2 within-subjects analysis of variance* was performed on quality scores as a function of EIPW (with, no tool) and task size (medium, moderate) as shown in Table 4.2.

Table 4.2 An ANOVA analysis of modeling quality

| | **QUALITY SCORE** |
|---|---|
| System (EIPW, no tool) | $F(1,20) = 97.512, p < 0.000$ |
| Task Size (medium, moderate) | $F(1,20) = 2.776, p < 0.111$ |
| System x Task Size | $F(1,20) = 1.085, p < 0.310$ |

Note: Significant Level < 0.05

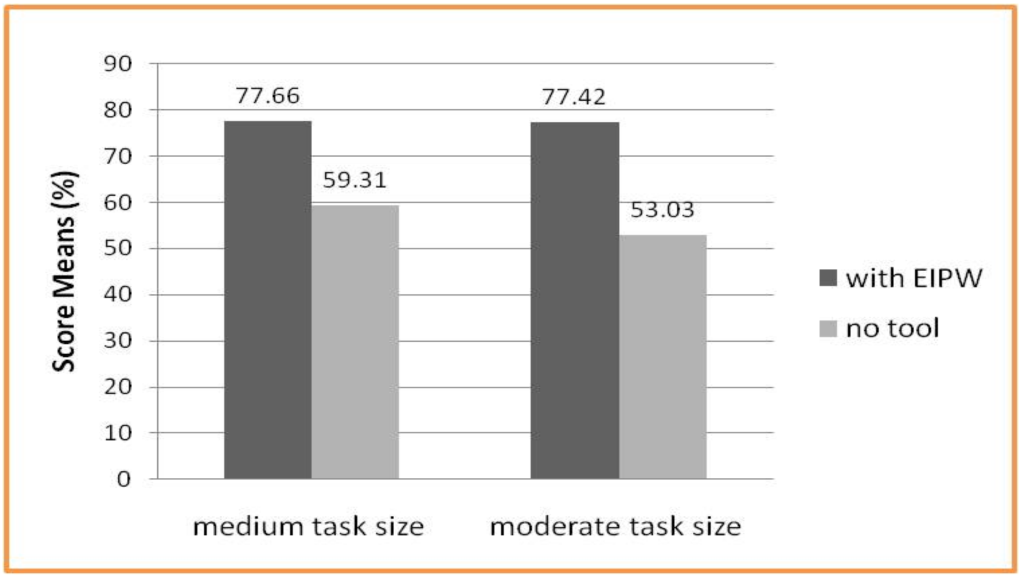


Figure 4.2 The plot of the mean quality scores (%)

From the calculated means shown in Figure 8, the conceptual models created by EIPW are better than those created by *no tool cases* for both task sizes. In Table 4.2, the results show that the main effect of **system** (with EIPW, no tool) is significant ($p < 0.00$). Therefore, this result supports our hypothesis (H1) that the EIPW helps novice designers create better conceptual models than they do without it. There is no significant main effect for **task size** ($p < 0.111$). It shows that the effect of **System x Task Size** is not significant ($p < 0.310$), which means there is no interaction between the system and the task size. We conclude that EIPW improves the novices' performance by 30.9% for the medium task size and 46.0% for the moderate task size.

## Test of Hypothesis 2: HBT

A *2x2 within-subjects analysis of variance* was performed on quality scores as a function of HBT (with, no tool) and task size (medium, moderate) as shown in Table 4.3.

Table 4.3 An ANOVA analysis of modeling quality

| | **QUALITY SCORE** |
|---|---|
| System (HBT, no tool) | F(1,19) = 25.69, $p < 0.000$ |
| Task Size (medium, moderate) | F(1,19) = 6.925, $p < 0.016$ |
| System x Task Size | F(1,19) = 0.132, $p < 0.720$ |

Note: Significant Level < 0.05

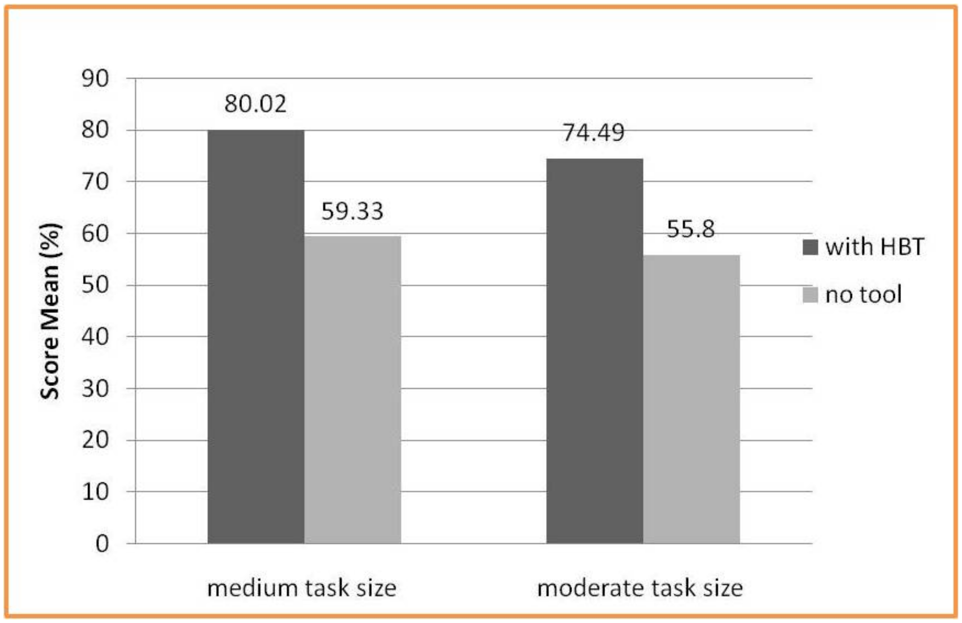


Figure 4.3 The plot of the mean quality scores (%)

From the calculated means shown in Figure 4.3, the conceptual models created by the HBT are better than those created by *no tool cases* for both task sizes. In Table 4.3, the results show that the main effect of **system** (with HBT, no tool) is significant (p < 0.00). Therefore, this results support our hypothesis (H2) that the HBT helps novice designers create better conceptual models than they do without it. There is significant main effect for **task size** (p < 0.016). However, it shows that the effect of **System x Task Size** is not significant (p < 0.720), which means there is no interaction between the system and the task size. We conclude that HBT improves the novices' performance by 34.9% for the medium task size and 33.5% for the moderate task size.

## Test of Hypothesis 3: EIPW & HBT

A *2x2 mixed model design with system as between-subject and task size as within-subject factors* was used. The two independent variables are **system** (with EIPW, with HBT) and the **task size** (medium, moderate). The dependent variable is the **quality score**. Since the aspects of within-subject factor are not used for analyzing this hypothesis, only the test of between-subject analysis is shown in Table 4.4.

Table 4.4 Tests of between-subjects effects with dependent variable QUALITY SCORE

| | **QUALITY SCORE** |
|---|---|
| System (EIPW, HBT) | F(1,39) = 0.004, p < 0.948 |

Note: Significant Level < 0.05

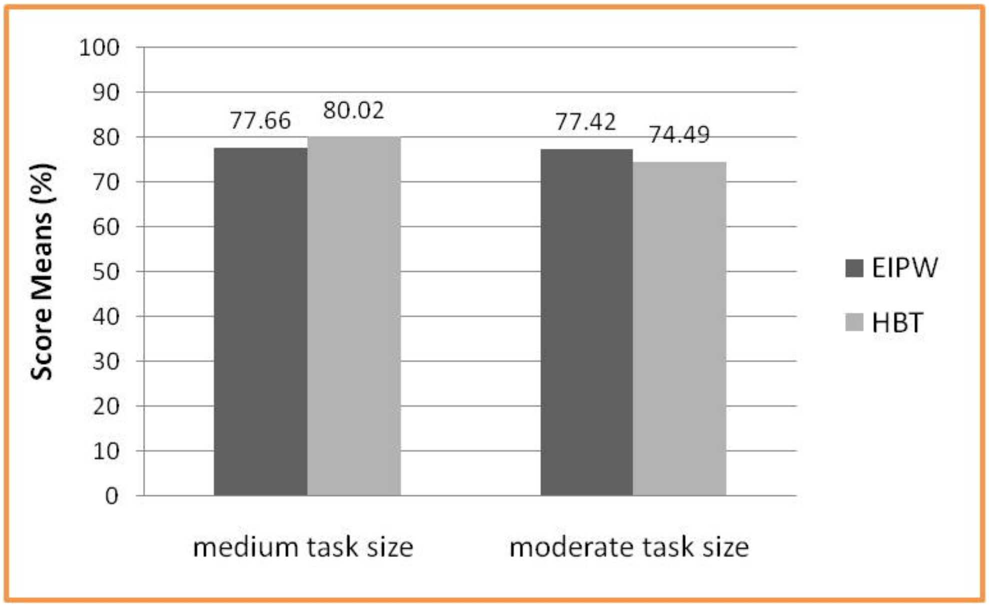


Figure 4.4 The plot of the mean quality scores (%)

In Table 4.4, the main effect of system is not significant ($p < 0.948$). So, this result supports our hypothesis (H3) that there is no significant difference between the two KBSs regarding the quality of the conceptual models. However, the mean scores of EIPW and HBT suggest that EIPW is better than HBT when the task size is moderate. On the other hand, HBT is slightly better than EIPW when the task size is smaller. This results show that the six domain independent modeling rules are effective in the small to medium task sizes.

## 4.5 Precision & Recall

Even though recall and precision (van Rijsbergen, 1979) are always used for evaluating information retrieval systems and also widely used in evaluating information extraction systems, Harmain and Gaizaukas (Harmain, 2003) first introduced precision and recall for evaluating conceptual data modeling systems. In any systems, both precision and recall should be close to 100% as possible. However, generally increasing in precision tends to decrease recall and vice versa. In this research, the definition of recall and precision are adopted as used by Harmain and Gaizaukas (2003).

*Recall* measures the completeness of the results developed by the system. The relevant information developed by the systems is compared with that developed by human analysts or answer key. The formula for calculating recall is:

$$Recall = \frac{N_{correct}}{N_{key}} \qquad \text{[Harmain \& Gaizauskas, 2003]}$$

Where $N_{correct}$ is the number of correct responses made by the system, and $N_{key}$ is the number of information elements in the answer key.

*Precision* measures the accuracy of the system (i.e. how much of the information produced by the system is correct). The formula for calculating precision is:

$$Precision = \frac{N_{correct}}{N_{correct}+N_{incorrect}} \qquad \text{[Harmain \& Gaizauskas, 2003]}$$

Where $N_{correct}$ is as above, and $N_{incorrect}$ is the incorrect responses made by the system.

The results of Recall and Precision of the performance of our proposed KBSs are shown in Table 4.5.

Table 4.5 Results of the performance of our KBSs

| **Tools or Systems** | **Recall** | **Precision** | **F-measure** |
|---|---|---|---|
| No tool | 56% | 74% | 64% |
| **EIPW** | **79%** | **93%** | **85%** |
| **HBT** | **84%** | **94%** | **89%** |

F-measure is the weighted average of the precision and recall.

It is questionable and inconsistent to use precision and recall to evaluate the performance of conceptual modeling tools or systems because there is no universal standard evaluation requirement corpus available. However, Table 4.5 provides the overall performance of our tools.

Since there is no standard evaluation of NL-based tools available, we cannot compare our tools with the previous tools or systems. However, Harmain & Gaizauskas (Harmain, 2003) claimed that other language processing technologies such as information retrieval systems, information extraction systems, and machine translation systems have found commercial applications with % precision and % recall well below this level (73% recall and 66% precision).

## 4.6 Limitations of the Research

The overall system performance has been evaluated in the previous chapter. In this section, the limitations of some of the system components are discussed.

1. This study has, so far, been carried out on one domain only, but it provides a theoretical background for research on other domains as well.
2. One characteristic of our KBSs is to integrate multiple modules and resources for the purpose of automating the process of conceptual data modeling. Some of the modules are designed for these KBSs while other are adopted from open source third party packages. Each component has some limitations.
   - The natural language processing (NLP) technique such as part of speech tagging (POS) technique cannot completely identify all of the information in the requirement specification. For example, the compound noun with hyphen cannot be identified as the whole one noun. However, the overall performance of the POS is quite effective and the % accuracy is 96.4.
   - The use of a general lexical knowledge resource, top noun categories and hypernym chains in WordNet, for automated entity identification process is novel and quite attractive. However, the top noun categories and hypernym chains in WordNet cannot perform entity categories completely because WordNet is a general knowledge resource and is not developed specifically for conceptual data modeling applications.
3. Outputs from the KBSs are individual relationships, not a combined ER diagram.
4. The evaluation of the completeness of the KBSs should be performed in many different categories. For example, empirical work either by using the systems in a number of real cases or by letting number of experts use and evaluate the tools.

# 5. CONCLUSION AND FUTURE WORK

Typically, conceptual data modeling has been considered as a creative activity, where human designers are indispensible. In this research, we have proposed methods that can improve the novice designers' performance and reduce the dependence on domain experts during the conceptual design process. Much research has been conducted in developing methodologies and guidelines to help the designers in conceptual database design. However, it would be useful if a designer has knowledge about an application domain in the form of repository of application-specific knowledge. Currently, building a repository of reusable artifacts involves explication of human developers' knowledge, which is a major obstacle in facilitating reuse of knowledge. To solve this problem, we proposed new types of reusable artifacts, called entity instance repository (EIR) and Relationship Instance Repository (RIR), which are repositories of Entity Instance Patterns (EIPs) and Relationship Instance Patterns (RIPs), respectively. These patterns can suggest what terms should appear in an application domain and how they are related to each terms. Our proposed artifacts are likely to be useful for conceptual designs in the following aspects: (1) they contain knowledge about a domain; (2) automatic generation of EIR and RIR overcomes a major problem of inefficient manual approaches that depend on experienced modeling designers and domain experts; and (3) they are domain-specific and therefore easier to understand and reuse. In this study, we provided a definition of the artifacts, and proposed the methodology for automatically generating repositories of domain artifacts.

We have implemented two knowledge-based data modeling tools: HBT and EIPW. HBT incorporates the six domain independent modeling rules, entity categories, and relationship instance repository. EIPW incorporates entity instance repository, entity categories, relationship instance repository, and WordNet. This step is an initial step to show how domain knowledge stored in the instance patterns can be used together with other modeling techniques.

The empirical results indicate that novice designers' performance increased by 30.9~46% when using EIPW, while the performance increased by 33.5~34.9 when using HBT, compared with the cases of no tools. The EIPW with EIR and RIR clearly helps the novice designers in creating better quality conceptual models. These results also imply that the use of EIR and RIR in EIPW is effective by providing us with a library of reusable patterns and by automating the process of finding the most appropriate one for certain situation. In addition, the results of HBT experiments show that the six domain independent modeling rules in HBT are effective in developing the quality conceptual models. They minimize the cognitive load on novices and ensure that the conceptual models are correct. This study shows that the six domain independent rules can be taught in a beginning database modeling class, and HBT can serve as a learning tool. It provides a smooth head-start to novices. In addition, RIR used in relationship identification process in both KBSs can ease the identification of relationships and solve the errors in conceptual models caused by the semantic mismatch in which not all real-world relationships can match the conceptual relationships.

The study has, so far, been carried out with one domain only, but it provides a theoretical background for research on other domains as well. However, it is necessary for the future research to advance the KBSs for supporting not only one but several different domains. Future work is to include more modeling rules required for automatic detection of modeling errors such as fan trap and chasm trap occurring in the conceptual data modeling process. We want to test the usability of the KBSs for different domains and subjects. The evaluation of the completeness of the KBSs have to be done by empirical work either by using the KBSs in a number of real cases or by letting a number of experts use and evaluate the KBS. We also plan to make both KBSs' interface modules to import the output schema into an ER diagram or a class diagram in graphical CASE tools. In addition, future research will consider the cognitive processes involved and mental representations in conceptual modeling design process. Cognitive science is a very useful reference discipline for gaining a deep understanding why people do things in a particular way (Siau, 1999).

# REFERENCES

Alexander, C. (1979). The Timeless Way of Building. New York: Oxford University Press.

Ambriola, V., Gervasi, V. (2006). On the systematic analysis of natural language requirements with circe. Automated Software Engineering, 13, 107-167.

Ambrosio, A. P., Métais, E., Meunier, J. N. (1995). The linguistic level of the KHEOPS CASE tool. Paper presented at the Proceedings of the 1st International Workshop on Applications of Natural Language to Data Bases (NLDB'95).

Anthony, S., Mellarkod, V. (2009). Data Modeling Patterns: A Method and Evaluation. Paper presented at the Proceedings of the Fifteenth Americas Conference on Information Systems, San Francisco, California.

Antony, S., Batra, D. (2002). CODASYS: a consulting tool for novice database designers. Paper presented at the SIGMIS Database.

Ba, S., Stallaert, J., Whinston, A.B. (2001). Research commentary: introducing a third dimension in information systems design -- the case for incentive alignment. Information Systems Research, 12(3), 337-355.

Batini, B., Demo, B., Leva, A. (1984). A methodology for conceptual schema design of office databases. Information System, 9, 251-263.

Batra, D. (2005). Conceptual Data Modeling Patterns: Representation and Validation. J. Database Manag., 16(2), 84-106.

Batra, D. (2007). Cognitive complexity in data modeling: causes and recommendations. Requir. Eng., 12(4), 231-244.

Batra, D., Antony, S. (1994). Novice errors in Conceptual database design. European Journal of Information Systems, 3(1), 57-69.

Blaha, M. (2010). Patterns of Data Modeling: CRC Press.

Bodart, F., Patel, A., Sim, M., Weber, R. (2001). Should Optional Properties Be Used in Conceptual Modeling? A Theory and Three Empirical Tests. Information Systems Research, 12(4), 384-405.

Boehm, B. (1981). Software Engineering Economics. Englewood Cliffs, NJ, USA: Prentice-Hall.

Buchholz, E., Cyriaks, H., Dsterhft, A., Mehlan, H., Thalheim, B. (1995). Applying a natural language dialogue tool for designing databases. Paper presented at the Proceedings of the first International Workshop on Applications of Natural Language to Databases (NLDB'95).

Burg, J., Van de Riet, P. (1998). Using Knowledge from WordNet for Conceptual Modeling. In Fellbaum, C. (ed.) WordNet: An Electromic Lexical Database. Cambridge, MA: MIT Press.

Castano, S., De Antonellis, V., Fugini, M., Pernici, B. (1998). Conceptual schema analysis: techniques and applications. ACM Transactions on Database Systems, 23(3), 286-333.

Castro, L., Baiao, F., Guizzardi, G. (2009). A Survey on Conceptual Modeling from a Linguistic Point of View: Technical Report, RelaTe-DIA

Chaiyasut, P., Shanks, G. (1994). Conceptual data modeling process: A study of novice and expert data modelers. Paper presented at the 1th International Conference on Object-Role Modeling, Australia, University of Queensland.

Chen, P. (1976). The Entity-Relationship Model: Toward A Unified View of Data. ACM Transactions on Database Systems, 1(1), 9-36.

Chen, P. (1983). English Sentence Structure and Entity-Relationship Diagram. Information Sciences, 1(1), 127-149.

Cheng, B. H. C., Atlee, J.M. (2007). Research directions in requirements engineering. Paper presented at the Proc. Future of Software Engineering FOSE 2007.

Chiang, H. (1993). Reverse engineering of relational databases: extraction of domain semantics. Ph.D Dissertation, University of Rochester, New York, USA.

Chiang, R., Barron, T., Storey, V. (1994). Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & Knowledge Engineering 12, 107-142.

Chisholm, R. (1996). A Realistic Theory of Categories - An Essay on Ontology: Cambridge University Press.

Choobineh, J., Lo, A. (2004). CABSYDD: Case-Based System for Database Design. Journal of Management Information Systems, 21(3), 242-253.

Choobineh, J., Mannino, M., Nunamaker, J. (1988). An expert database design system based on analysis of forms. IEEE Transaction on Software Engineering, 14, 242-253.

Coad, P., North, D., Mayfield, M. (1995). Object Models – Strategies, Pattern, & Applications.: Englewood Cliffs: Yourdon Press.

Conesa, J., Storey, V., Sugumaran, V . (2007). Experiences Using the ResearchCyc Upper level Ontology. In Z. K. e. a. (Eds.) (Ed.), NLDB 2007, LNCS 4592 (pp. 143-155): Springer-Verlag.

Conesa, J., Storey, V., Sugumaran, V. . (2010). Usability of Upper level ontologies: The case of ResearchSyc. Data & Knowledge Engineering, 69(4).

Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A. . (2003). Methodologies, Tools and Languages for Building Ontologies: Where is their meeting point? Data & Knowledge Engineering, 46, 41–64.

Currim, S. (2008). Towards Improving Conceptual Modeling: an examination of common errors and their underlying reasons. Ph.D Dissertation, University of Arrizona.

Dehne, F., Steuten, A., R.P. van de Riet. (2001). WordNet++: "A lexicon for the Color-X method. Data and Knowledge Engineering, 38(1), 3-29.

Dey, D., Storey, V., Barron, T. (1999). Improving database design through the analysis of relationships. ACM Transactions on database systems, 24(4), 453-486.

Du, S. (2008). On the Use of Natural Language Processing for Automated Conceptual Data Modeling. Ph.D Dissertation, University of Pittsburgh.

Dullea, J., Song, I., Lamprou, I. (2003). An Analysis of Structural Validity in ER Modeling. Data and Knowledge Engineering, 47(2), 167-205.

Dumpala, S. R., Arora, S.K. (1981). Schema Translation Using the Entity Relationship Approac. In P. P. E. Chen (Ed.), Entity-Relationship Approach to Information Modeling and Analysis, ER Institute (pp. 339-360).

Eick, C. F., & Lockemann, P. C. (1985). Acquisition of Terminology Knowledge Using Database Design Techniques. Paper presented at the Proceedings ACM SIGMOD conference, Austin, USA.

El-Ghalayini, H., Odeh, M., McClatchey, R. (2006). Engineering Conceptual Data Models from Domain Ontologies: A Critical Evaluation. International Journal of Information Technology and Web Engineering, 2(1), 57-70.

Elmasri, R., Nevathe, S. (2004). Fundamentals of Database Systems (3rd ed). Redwood City, CA: The Benjamin/Cummings Publishing Co., Inc.

Embley, D. (2004). Toward Semantic Understanding an Approach Based on Information On Information Extraction Ontologies. Paper presented at the Proceedings of the Fourteenth Australian Database Conference, Denedin, New Zealand.

Evermann, J., Wand, Y. (2001). Towards Ontologically-Based Semantics for UML Constructs. In H. S. Kunii, Jajodia, S., Solvberg, A., (eds) (Ed.), Conceptual Modeling-ER 2001, Lecture Notes in Computer Science (pp. 341-354): Springer.

Fahrner, C., Vossen, G. (1995). A survey of database design transformations based on the Entity- Relationship model. Data Knowledge Eng., 15(3), 213-250.

Fayad, M., Schmidt, D., Johnson, R.  (1997). Object-oriented Application Frameworks: Problem and Perspectives. NY: Willy.

Fellbaum, C. (1998). WordNet: An Electronic Lexical Database. Fellbaum, C. (Ed.). Cambridge, MA: MIT Press.

Fernandez, E. B., Yuan, X. (2000). Semantic Analysis Patterns. Paper presented at the Procs. of the 19th Int. Conf. on Conceptual Modeling (ER2000).

Fonseca, F., Martin, J. (2007). Learning the Differences Between Ontologies and Conceptual Schemas Through Ontology-Driven Information Systems. JAIS - Journal of the Association for Information Systems - Special Issue on Ontologies in the Context of IS 8(2), 129–142.

Fowler, M. (1997). Analysis Patterns: Reusable Object Models. Menlo Park, CA, USA: Addison Wesley.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995). Design Patterns:Elements of Reusable Object-Oriented Software: Addison Wesley.

Gentner, D., Medina, J.  (1998). Similarity and the development of rules.  Cognition,  65, 263—297.

Gogolla, M., Hohenstein, U. (1991). Towards a semantic view of an extended entity-relationship model.  ACM Transactions on Database Systems, 16(3), 369-416.

Gomez, F., Segami, C., Delaune, C. (1999). A System for the Semi-Automatic Generation of E-R Models from Natural Language Specifications. Knowledge and Data Engineering, 29, 57-81.

Guarino, N., Welty, C. (2004). An overview of OntoClean. In S. Staab, Studer, R. (eds.) (Ed.), Handbook on Ontologies (pp. 151-159): Springer Verlag.

Han, T. (2002). Automating Reuse for Systems Design. Ph.D Dissertation., Georgia State University.

Han, T., Purao, S., Storey, V.  (2008). Generating large-scale repositories of reusable artifacts for conceptual design of information systems. Decision Support Systems, 45, 665-680.

Harmain, M., Gaizauskas, R.  (2003). CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Anaysis. Automated Software Engineering, 10(2), 157-181.

Hartmann, S., Link, S. (2007). English Sentence Structures and EER modeling. Paper presented at the 4th Asia-Pacific Conference on Conceptual Modeling.

Hay, D. C. (1996). Data model patterns: Conventions of Thought. New York: Dorset House Publishing.

Hitchman, S. (2003). An Interpretive Study of How Practitioners Use Entity-Relationship Modelling in a Ternary Relationship Situation. Communications of the Association for Information Systems, 11(26).

Hoffer, J., Prescott, M., Mcfadden, F. (2004). Modern database management (7 ed.). Upper Saddle River, New Jersey: Pearson Prentice Hall.

Johannesson, P., Kalman, K.  (1989). A method for translating relational schemas into conceptual schemas. Paper presented at the Proc. Eighth Int. Conf. on Entity-Relationship Approach.

Johannesson, P., & Wohed, P. (1999). The deontic patterns-a framework for domain analysis in information systems design. Data & Knowledge Engineering, 31.

Johnson, R., Foote, B.  (1988). Designing reusable classes. Journal of Object-Oriented Programming, 1(2), 22-35.

Kankanhalli, A., Tan, B., Wei, K. (2005). Contributing knowledge to electronic knowledge repositories: an emprical investigation. MIS Quarterly, 29, 113-143.

Kim, N., Lee, S., Moon, S.  (2008). Formalized Entity Extraction Methodology for Changeable Business Requirements. Journal of Information Science and Engineering, 24, 649-671.

Kim, Y., March, S.  (1995). Comparing data modeling formalisms. Communications of the ACM, 38(6), 103-115.

Kimball, R., Ross, M. (2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Data Modeling (2nd  ed.). New York: John Wiley & Sons, Inc.

Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (3rd  ed.): Prentice Hall, Englewood Cliffs, New Jersey.

Lenat, D. B. (1995). CYC: a large-scale investment in knowledge infrastructure. . Paper presented at the Communication ACM

Liao, C., Palvia, P. (2000). The impact of data models and task complexity on end-user performance: an experimental investigation. Journal of Human-Computer Studies, 41(4), 212-218.

Markus, M. L. (2001). Towards a theory of knowledge reuse: types of knowledge reuse situations and factors in reuse success. Journal of Management Information Systems, 18(1), 57–94.

Mascardi, V., Cordì, V., Rosso, P. (2007). A Comparison of Upper Ontologies Technical Report DISI-TR-06-2.

Métais, E. (2002). Enhancing information systems management with natural language processing techniques. Data Knowl. Eng, 41(2), 247-272.

Meziane, F., & Vadera, S. (2004). Obtaining E-R Diagrams Semi-Automatically From Natural Lnaguage Specifications. Paper presented at the Proceedings of the 6th International Conference of Enterprise Information Systems, ICEIS 2004, Portugal.

Mich, L., Franch, M., Inverardi, P. (2004). Market research for requirements analysis using linguistic tools. Requirements Eng., 40-56.

Mich, L., & Garigliano, R. (1999). The NL-OOPS project: object oriented modeling using the natural language processing system LOLITA. Paper presented at the Proceedings of the 4th International Conference on the Applications of Natural Language to Information Systems (NLDB'99).

Mich, L., Garigliano, R. (2002 ). NL-OOPS: A Requirements Analysis tool based on Natural Language Processing. Paper presented at the Proc. 3rd Int. Conf. On Data Mining 2002, Bologna.

Milton, S., Kazmierczak, E. (2004). An Ontology for Data Modeling Languages: A Study Using a Common-Sense Realistic Ontology. Journal of Database Management, 15(2).

Miyoshi, H., Sugiyama, K., Kobayashi, M., Ogino, T. (1996). An Overview of the EDR Electronic Dictionary and the Current Status of Its Utilization. Paper presented at the Proceedings of the 16th International Conference on Computational Linguistics.

Moody, D. (2004). Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. ADBIS 2004, 129-143.

Moody, D. (2005). Theoretical and Practical Issues in evaluating the quality of conceptual models: Current State and Future Directions. Data & Knowledge Engineering, 55, 243-276.

Moody, D., Shanks, G. (1994). What makes a good data model? Evaluating the quality of entity-relationship models. Paper presented at the 13th International Conference on the Entity-Relationship Approach, Manchester.

Moody, D. L. (1998). Metrics for Evaluating the Quality of Entity Relationship Models. ER 1998, 211-225.

Moody, D. L., Shanks, G.G. (2003). Improving the quality of data models: empirical validation of a quality management framework. Inf. Syst, 28(6), 619-650.

Neill, C., Laplante, P. (2003). Requirement engineering: the state of the practice. IEEE Software, 20(6), 40-45.

Omar, N. (2004). Heuristics-based Entity-Relationship Modeling through Natural Language Processing. Ph.D Dissertation, University of Ulster.

Omar, N., Hanna, P., Mc Kevitt, P. (2004). Heuristics-Based Entity-Relationship Modeling through Natural Language Processing. Paper presented at the Proc. Of the fifteen Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04).

Orlikowski, W. J. (1993). Learning from notes: organizational issues in groupware implementation. Information Society, 9(3), 237–251.

Osborne, M., MacNish, C. K. (1996). Processing natural language software requirement specifications. Paper presented at the Second International Conference on Requirements Engineering (ICRE'96).

Overmyer, S., Lavoie, B., Rambow, O. (2001). Conceptual Modeling through Linguistic Analysis Using LIDA. ICSE 2001, 401-410.

Paek, Y.-K., Seo, J., Kim, G.-C. (1996). An expert system with case-based reasoning for database schema design. Decision Support Systems, 18(1), 83-95.

Parson, J., Saunders, C. (2004). Cognitive heuristics in software engineering: applying and extending anchoring and adjustment to artifact reuse. IEEE Trans. Software Engineering, 30(12), 873-888.

Pree, W. (1994). Design Patterns for Object-Oriented Software Development: Addison-Wesley.

Prieto-Diaz, R. (1987). Domain Analysis for reusability. Paper presented at the COMPSAC'87, Tokyo, Japan.

Purao, S. (1998). APSARA: A tool to automate system design via intelligent pattern retrieval and synthesis. Database Advance Information Systems, 29(4), 45-57.

Purao, S., Storey, V. C. (2005). A multi-layered ontology for comparing relationship semantics in conceptual models of databases. J. Applied Ontology, 1(1), 117–139.

Purao, S., Storey, V., Han, T. (2003). Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning. Information Systems Research, 14(3), 269-290.

Rumbaugh, J., Blaha, M., Premerlani, W. (1991). Object-Oriented Modeling and Design: Prentice-Hall.

Sherif, K. (2002). Domain engineering for developing software repositories: a case study. Decision Support Systems, 33(1), 55-69.

Shoval, P., Shiran, S. (1997). Entity-Relationship and Object-Oriented Data Modeling—An experimental comparison of design quality. Data & Knowledge Engineering, 21(3), 297-315.

Siau, K. (1999). Information Modeling and Method Engineering: A Psychological Perspective. J. Database Manag., 10(4), 44-50.

Silverston, L. (2001). The Data Model Resource Book Revised Edition Volume 2: John Willey & Sons Inc.

Simsion, G. (2007). Data Modeling Theory and Practice: Technique Publications, LLC.

Soares, A., Fonseca, F. (2007). Ontology-Driven Information Systems: At Develop time. International Journal of Computers, Systems and Signals, 8(2).

Song, I.-Y., Evans, M., Park, E. (1995). A Comparative Analysis of Entity-Relationship Diagrams. Journal of Computer and Software Engineering, 3(4), 427-459.

Song, I.-Y., Jones, T . (1993). Analysis of binary relationships within ternary relationships in ER modeling. Paper presented at the Proc. Of the 12th International Conference on Entity-Relationship Approach, Dallas, TX.

Song, I.-Y., Yano. K., Trujillo, J., Lujan-Mora, S. . (2004). A Taxonomic Class Modeling Methodology for Object-Oriented Analysis. In Information Modeling Methods and

Methodologies. In T. H. J Krostige, K. Siau (Ed.), Advanced Topics in Databases Series, Ed. (pp. 216-240): Idea Group Publishing.

Storey, V. C. (1993a). A Selective Survey of the use of Artificial Intelligence for Database Design Systems. Data & Knowledge Engineering, 11, 61-102.

Storey, V. C. (1993b). Understanding semantic relationships. VLDB Journal 2, 455-488.

STOREY, V. C. (2005). Classifying and comparing relationships in conceptual modeling. IEEE Trans. Knowl. Data Engin., 17(11), 1-13.

Storey, V. C., Chiang, R., Goldstein, R., Dey, D., Sundaresan, S. (1997). Database design with common sense business reasoning and learning. ACM Transactions on Database Systems, 22(4), 471-512.

Sugumaran, V., Storey, V. (2006). The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Trans. Database System, 31(3), 1064-1094.

Szyperski, C. (1998). Component Software: Beyond Object-Oriented Programming: Addison-Wesley.

Teorey, T., Yang, D., Fry, J. (1986). A logical design methodology for relational databases using the extended entity-relationship model. ACM Computing Surveys, 18, 197-222.

Thalheim, B. (2000). Entity-Relationship Modeling: Foundations of Database Technology. Berlin Heidelberg: Springer Verlag.

Thonggoom, O., Song, I.-Y., An, Y. (2011). EIPW: A Knowledge-based Database Modeling Tool. Paper presented at the CAiSE Workshops, London, England.

Tjoa, A., Berger, L. (1993). Transformations of requirements specifications expressed in natural language into an EER model. Paper presented at the Proceedings of the 12th International Conference on the Entity-Relationship Approach: Entity-Relationship Approach.

Topi, H., Ramesh, V. (2002). Human factors research on data modeling: a review of prior research, an extended framework and future research directions. J Database Management, 13, 3-15.

Tseng, F., Chen, C. (2008). Enriching the class diagram concepts to capture natural language semantics for database access. Data & Knowledge Engineering, 67(1), 1-29.

Tseng, F. S. C., Chen, A.L.P., Yang, W. (1992). On Mapping Natural Language Constructs into Relational Algebra Through E-R Representation. Data & Knowledge Engineering (9), 97-118.

van Rijsbergen, C. J. (1979). Information Retrieval (2 ed.). London.

Wand, Y., Storey, V.C., Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. ACM Transactions on Database Systems, 24(4), 494-528.

Wand, Y., Weber, R. (1988). An Ontological Analysis of Some Fundamental Information Systems Concepts. Paper presented at the 9th International Conference on Information Systems, Minneapolis, Minnesota.

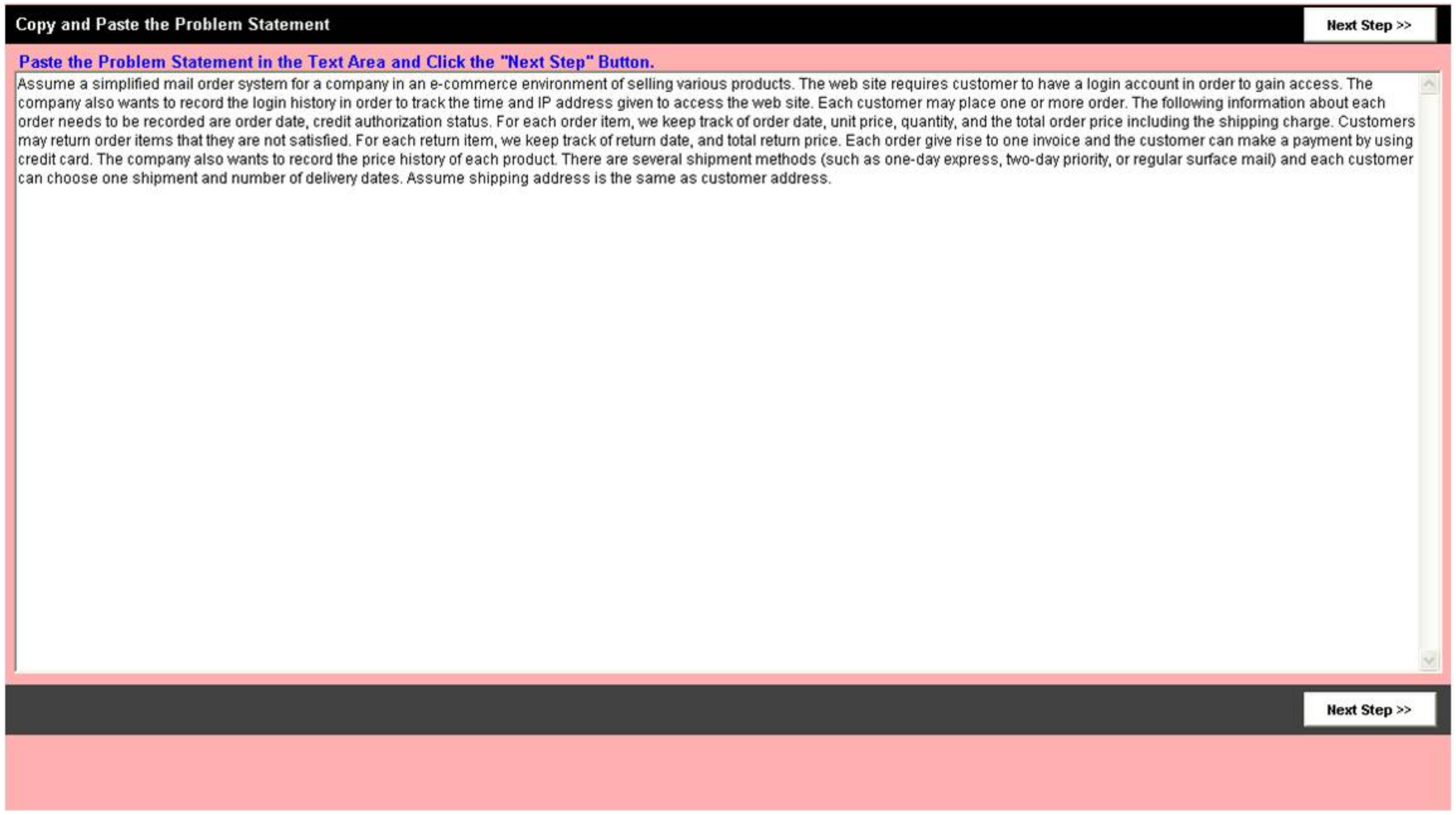
Wand, Y., Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. . Journal of Information Systems, 3(4).

Wand, Y., Weber, R. (1995). Theoretical foundations for conceptual modeling in information systems development. Decision Support Systems, 15(4), 285-305.

Wohed, P. (2000). Conceptual patterns for reuse in information systems analysis. Paper presented at the Proc. 12th Internat. Conf. Adv. Inform. Systems.

# Appendix A: The user interfaces of EIPW

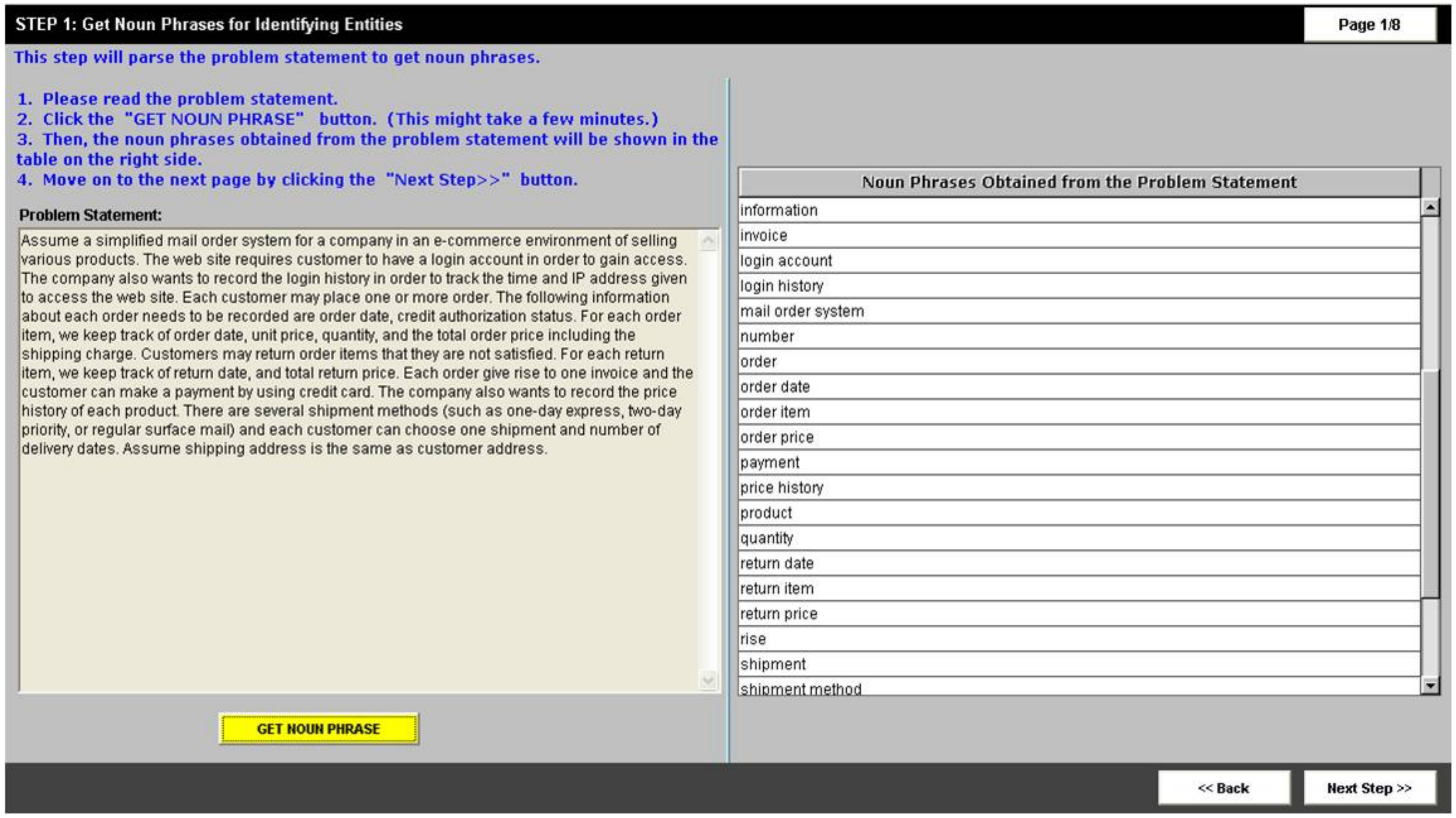
Initial Step: Begin with copy and paste a requirement specification in the text box.


Copy and Paste the Problem Statement
Next Step >>
Paste the Problem Statement in the Text Area and Click the "Next Step" Button.
Next Step >>

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

Figure A.1 A Screenshot of Initial Step in EIPW

Step 1: Get all the noun phrases from the requirement specification by applying POS.



STEP 1: Get Noun Phrases for Identifying Entities

Page 1/8

This step will parse the problem statement to get noun phrases.

1. Please read the problem statement.
2. Click the "GET NOUN PHRASE" button. (This might take a few minutes.)
3. Then, the noun phrases obtained from the problem statement will be shown in the table on the right side.
4. Move on to the next page by clicking the "Next Step>>" button.

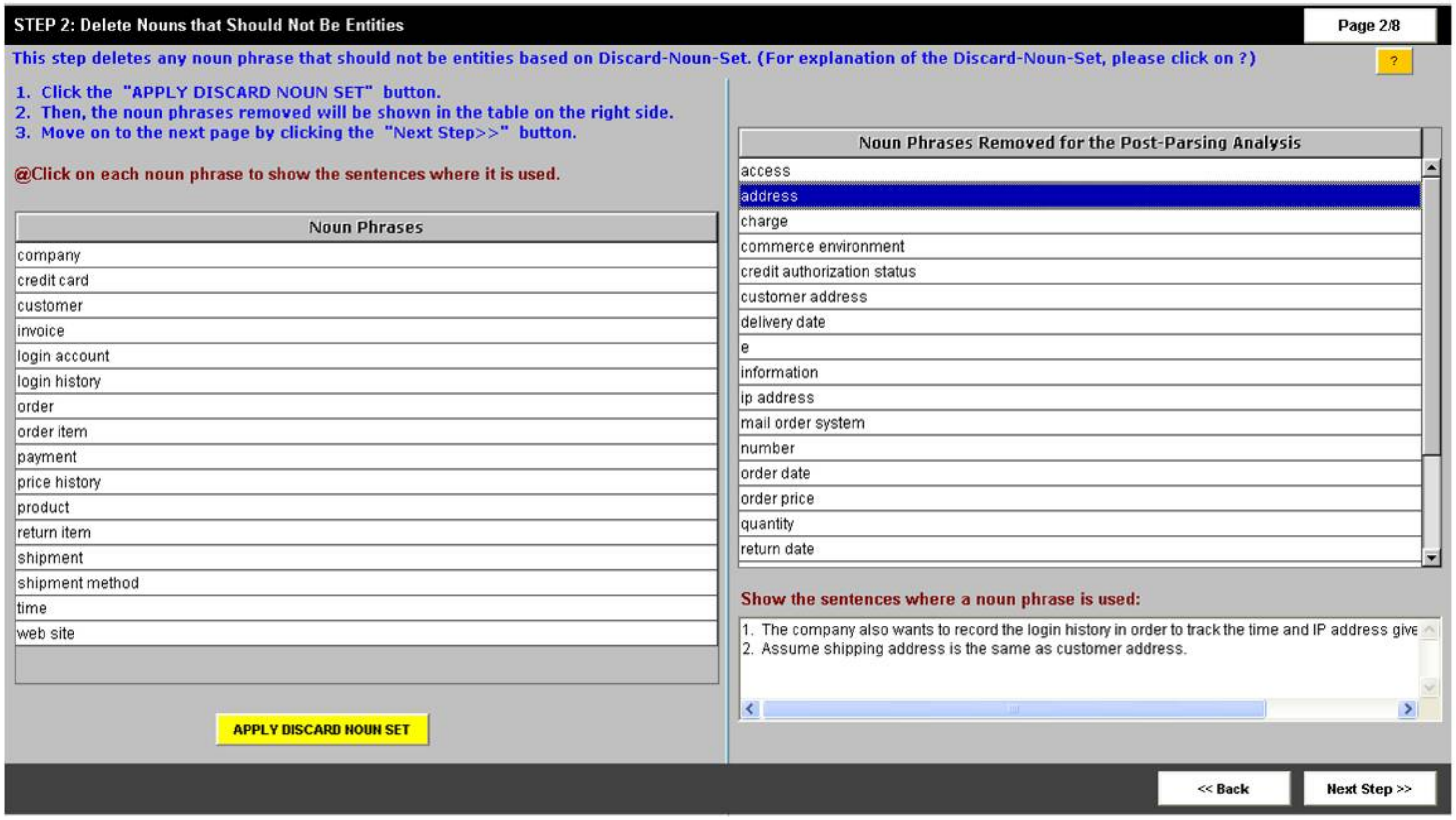
**Problem Statement:**

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

| Noun Phrases Obtained from the Problem Statement |
|---|
| information |
| invoice |
| login account |
| login history |
| mail order system |
| number |
| order |
| order date |
| order item |
| order price |
| payment |
| price history |
| product |
| quantity |
| return date |
| return item |
| return price |
| rise |
| shipment |
| shipment method |

Figure A.2 A Screenshot of Step 1 in EIPW

Step2: Test the discard noun set for facilitating the post-parsing analysis.



STEP 2: Delete Nouns that Should Not Be Entities

Page 2/8

This step deletes any noun phrase that should not be entities based on Discard-Noun-Set. (For explanation of the Discard-Noun-Set, please click on ?)

1. Click the "APPLY DISCARD NOUN SET" button.
2. Then, the noun phrases removed will be shown in the table on the right side.
3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase to show the sentences where it is used.

| Noun Phrases |
|---|
| company |
| credit card |
| customer |
| invoice |
| login account |
| login history |
| order |
| order item |
| payment |
| price history |
| product |
| return item |
| shipment |
| shipment method |
| time |
| web site |

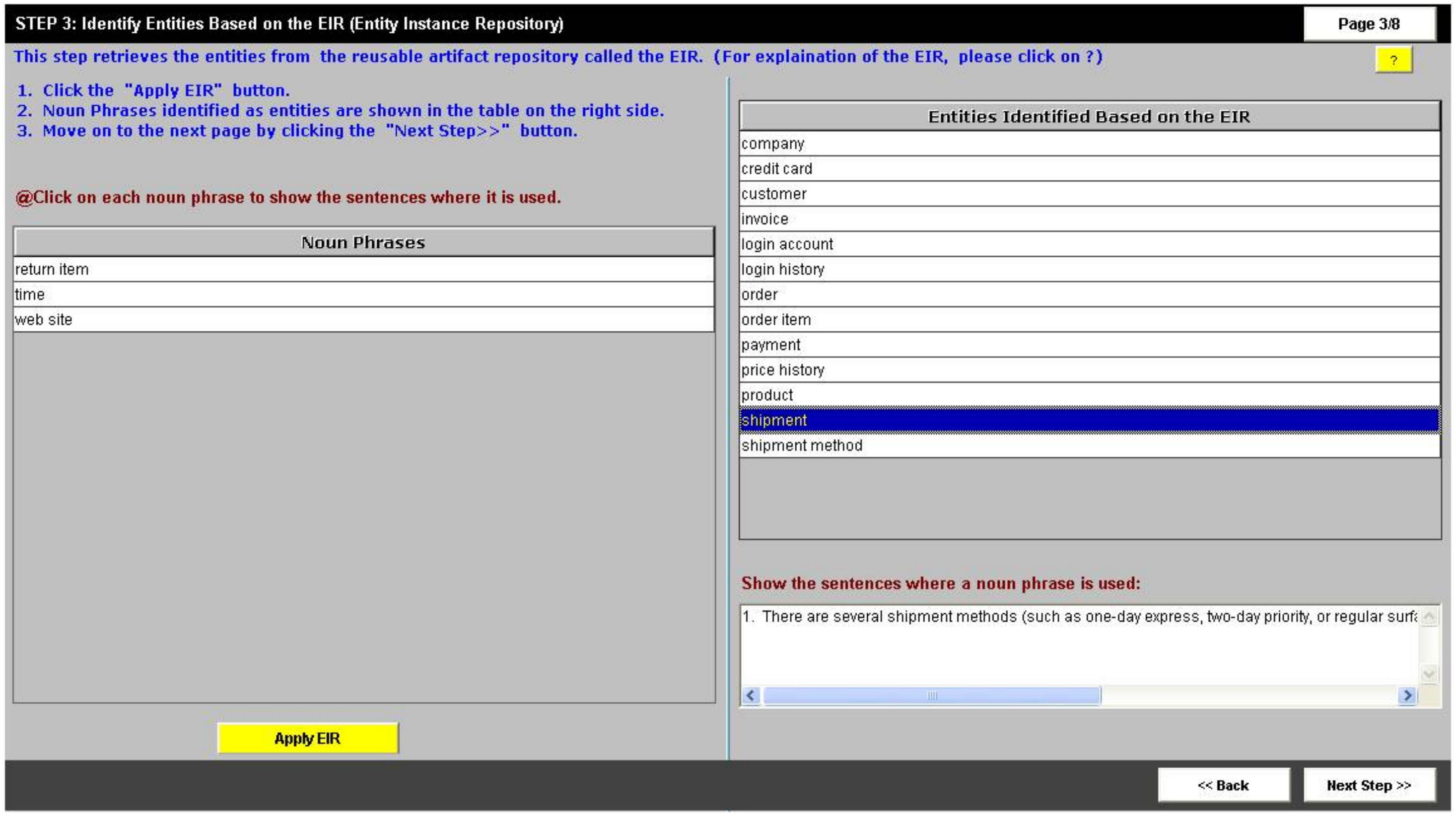
| Noun Phrases Removed for the Post-Parsing Analysis |
|---|
| access |
| address |
| charge |
| commerce environment |
| credit authorization status |
| customer address |
| delivery date |
| e |
| information |
| ip address |
| mail order system |
| number |
| order date |
| order price |
| quantity |
| return date |

Show the sentences where a noun phrase is used:

1. The company also wants to record the login history in order to track the time and IP address give
2. Assume shipping address is the same as customer address.

Figure A.3 A Screenshot of Step 2 in EIPW

Step 3: Identify entities based on EIR (Entity Instance Repository). And out of the entity names in EIR, identify synonyms from WordNet.



STEP 3: Identify Entities Based on the EIR (Entity Instance Repository)

Page 3/8

This step retrieves the entities from the reusable artifact repository called the EIR. (For explaination of the EIR, please click on ?)

1. Click the "Apply EIR" button.
2. Noun Phrases identified as entities are shown in the table on the right side.
3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase to show the sentences where it is used.

| Noun Phrases |
|---|
| return item |
| time |
| web site |

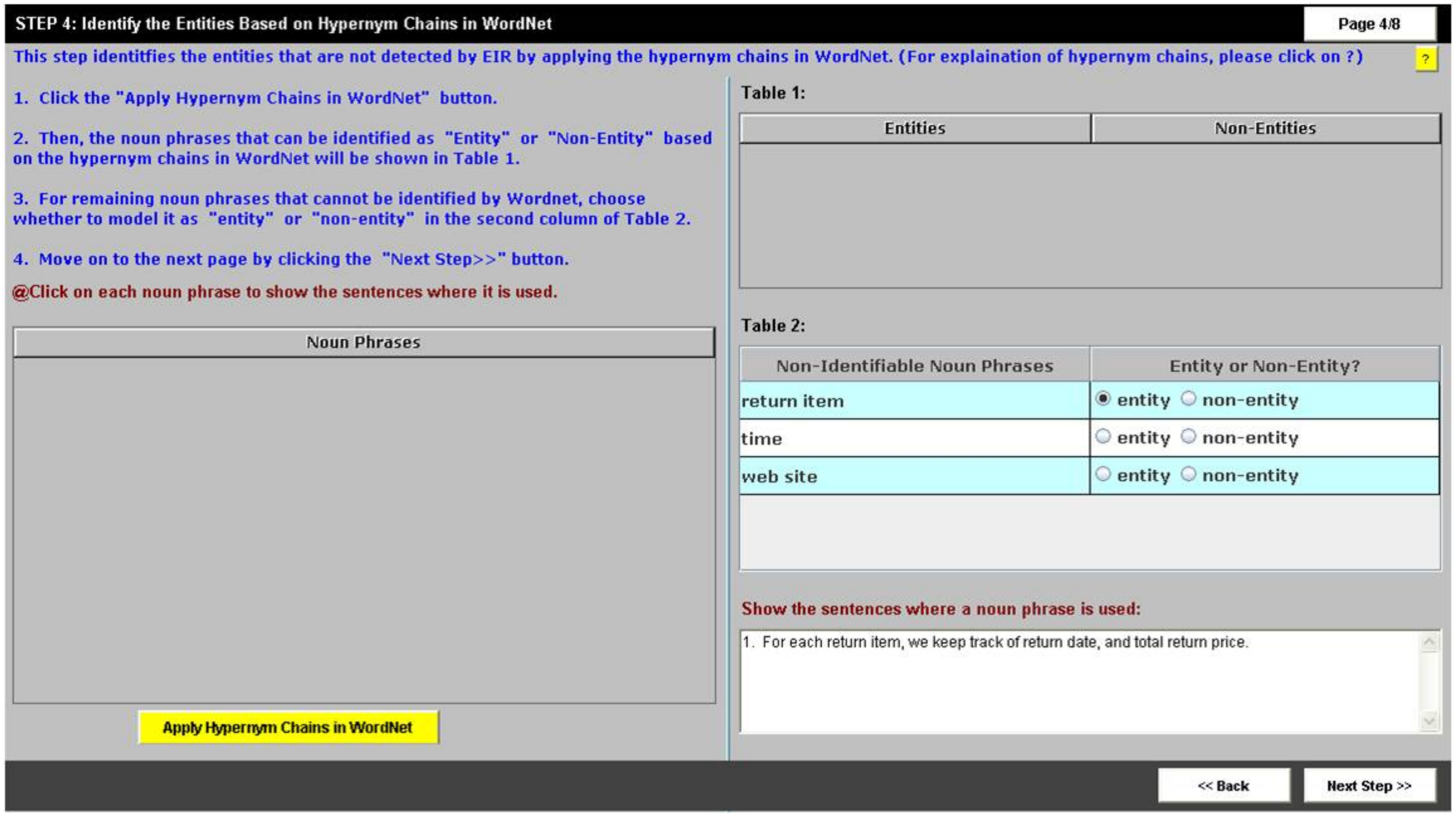
| Entities Identified Based on the EIR |
|---|
| company |
| credit card |
| customer |
| invoice |
| login account |
| login history |
| order |
| order item |
| payment |
| price history |
| product |
| shipment |
| shipment method |

Show the sentences where a noun phrase is used:

1. There are several shipment methods (such as one-day express, two-day priority, or regular surf

Figure A.4 A Screenshot of Step 3 in EIPW

Step 4: Identify entities based on top noun categories and hypernym chains in WordNet.


STEP 4: Identify the Entities Based on Hypernym Chains in WordNet
Page 4/8
?
Noun Phrases
Apply Hypernym Chains in WordNet
<< Back
Next Step >>

This step identitfies the entities that are not detected by EIR by applying the hypernym chains in WordNet. (For explaination of hypernym chains, please click on ?)

1. Click the "Apply Hypernym Chains in WordNet" button.

2. Then, the noun phrases that can be identified as "Entity" or "Non-Entity" based on the hypernym chains in WordNet will be shown in Table 1.

3. For remaining noun phrases that cannot be identified by Wordnet, choose whether to model it as "entity" or "non-entity" in the second column of Table 2.

4. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase to show the sentences where it is used.

Table 1:

| Entities | Non-Entities |
|---|---|

Table 2:

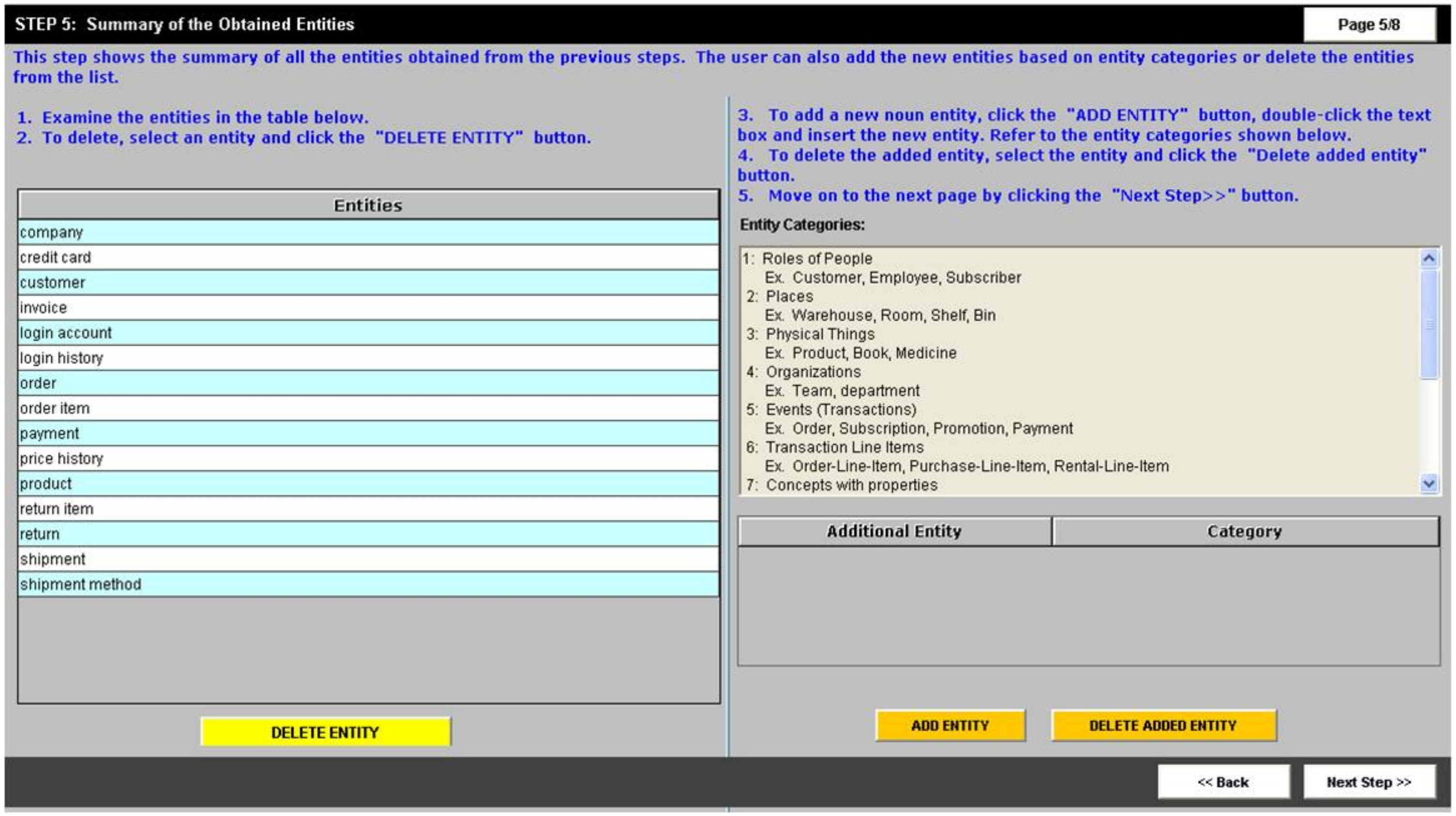
| Non-Identifiable Noun Phrases | Entity or Non-Entity? |
|---|---|
| return item | [✓ entity] non-entity |
| time | entity non-entity |
| web site | entity non-entity |

Show the sentences where a noun phrase is used:

1. For each return item, we keep track of return date, and total return price.

Figure A.5 Screenshot of Step 4 in EIPW

Step 5: Identify entities based on top noun categories and hypernym chains in WordNet.



STEP 5: Summary of the Obtained Entities

Page 5/8

This step shows the summary of all the entities obtained from the previous steps. The user can also add the new entities based on entity categories or delete the entities from the list.

1. Examine the entities in the table below.
2. To delete, select an entity and click the "DELETE ENTITY" button.

| Entities |
|---|
| company |
| credit card |
| customer |
| invoice |
| login account |
| login history |
| order |
| order item |
| payment |
| price history |
| product |
| return item |
| return |
| shipment |
| shipment method |

3. To add a new noun entity, click the "ADD ENTITY" button, double-click the text box and insert the new entity. Refer to the entity categories shown below.
4. To delete the added entity, select the entity and click the "Delete added entity" button.
5. Move on to the next page by clicking the "Next Step>>" button.

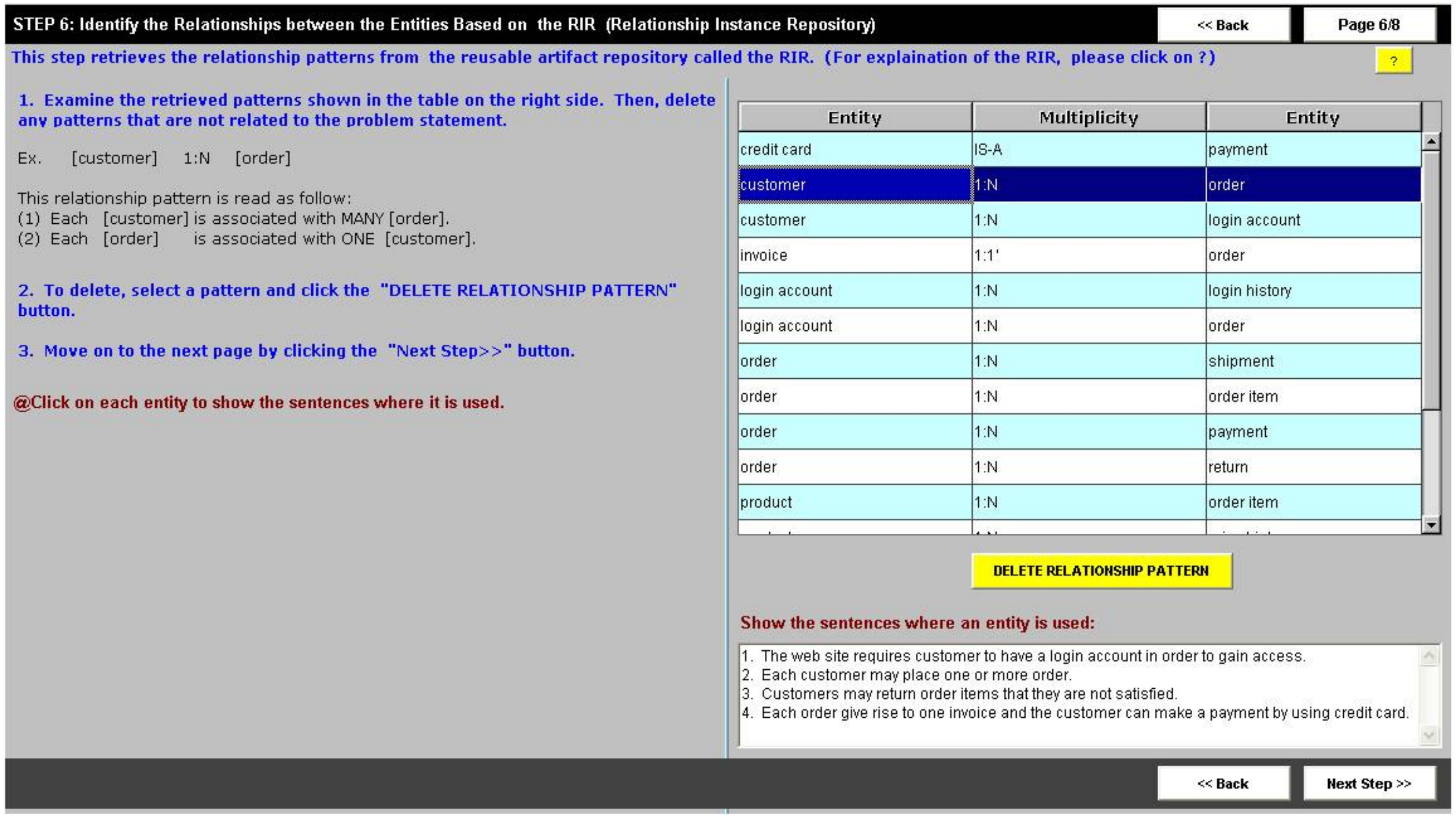
Entity Categories:

1: Roles of People
Ex. Customer, Employee, Subscriber
2: Places
Ex. Warehouse, Room, Shelf, Bin
3: Physical Things
Ex. Product, Book, Medicine
4: Organizations
Ex. Team, department
5: Events (Transactions)
Ex. Order, Subscription, Promotion, Payment
6: Transaction Line Items
Ex. Order-Line-Item, Purchase-Line-Item, Rental-Line-Item
7: Concepts with properties

| Additional Entity | Category |
|---|---|

Figure A.6 A Screenshot of Step 4 in EIPW

Step 6: identify relationships based on RIR (Relationship Instance Repository).


STEP 6: Identify the Relationships between the Entities Based on the RIR (Relationship Instance Repository)

This step retrieves the relationship patterns from the reusable artifact repository called the RIR. (For explaination of the RIR, please click on ?)

1. Examine the retrieved patterns shown in the table on the right side. Then, delete any patterns that are not related to the problem statement.

Ex. [customer] 1:N [order]

This relationship pattern is read as follow:
(1) Each [customer] is associated with MANY [order].
(2) Each [order] is associated with ONE [customer].

2. To delete, select a pattern and click the "DELETE RELATIONSHIP PATTERN" button.

3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each entity to show the sentences where it is used.

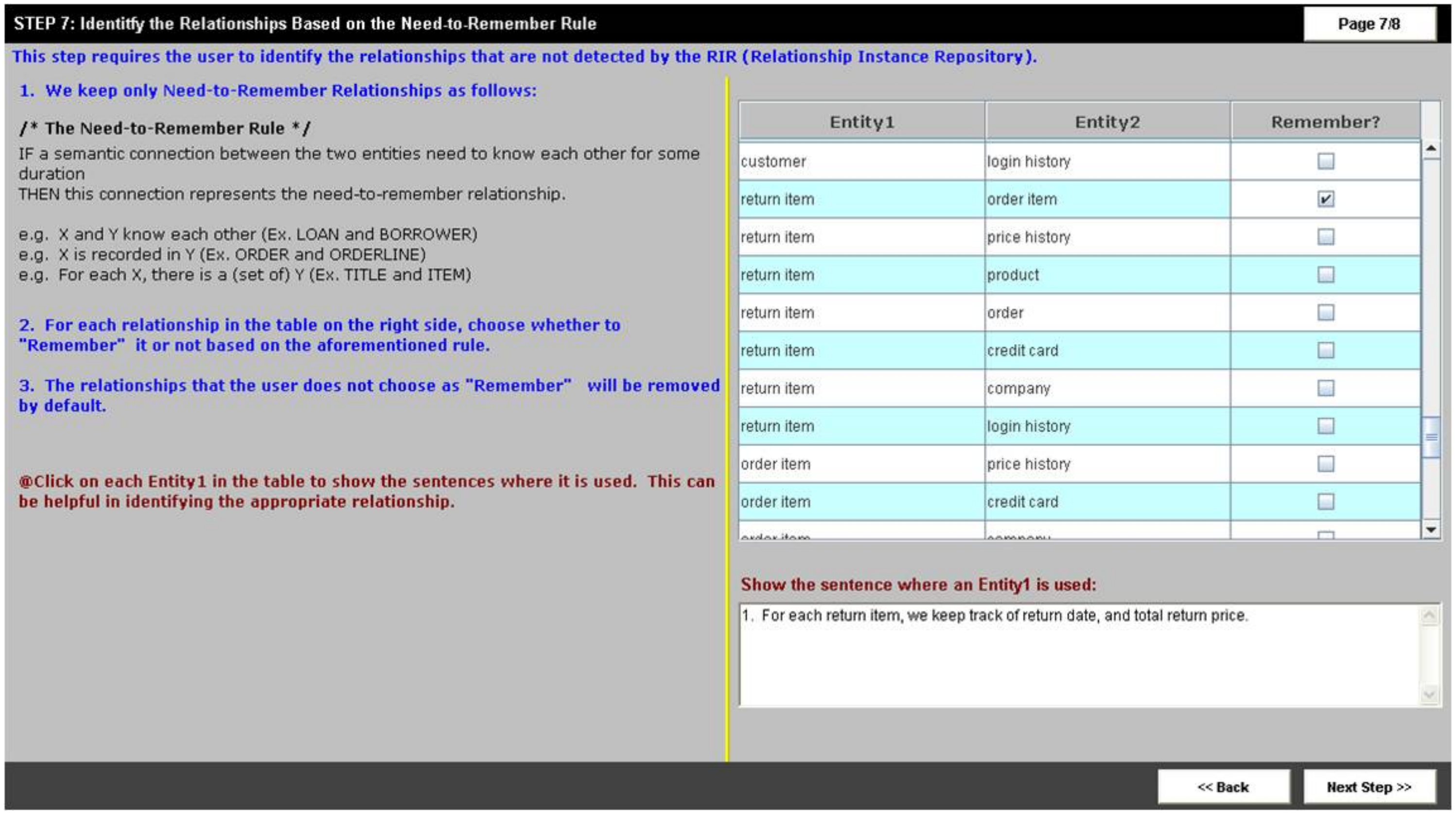
| Entity | Multiplicity | Entity |
|---|---|---|
| credit card | IS-A | payment |
| customer | 1:N | order |
| customer | 1:N | login account |
| invoice | 1:1' | order |
| login account | 1:N | login history |
| login account | 1:N | order |
| order | 1:N | shipment |
| order | 1:N | order item |
| order | 1:N | payment |
| order | 1:N | return |
| product | 1:N | order item |

Show the sentences where an entity is used:

1. The web site requires customer to have a login account in order to gain access.
2. Each customer may place one or more order.
3. Customers may return order items that they are not satisfied.
4. Each order give rise to one invoice and the customer can make a payment by using credit card.

Figure A.7 A Screenshot of Step 6 in EIPW

Step 7: Identify relationships, which are not detected by RIR, by applying a modeling rule.



STEP 7: Identify the Relationships Based on the Need-to-Remember Rule

Page 7/8

This step requires the user to identify the relationships that are not detected by the RIR (Relationship Instance Repository).

1. We keep only Need-to-Remember Relationships as follows:

/* The Need-to-Remember Rule */
IF a semantic connection between the two entities need to know each other for some duration
THEN this connection represents the need-to-remember relationship.

e.g. X and Y know each other (Ex. LOAN and BORROWER)
e.g. X is recorded in Y (Ex. ORDER and ORDERLINE)
e.g. For each X, there is a (set of) Y (Ex. TITLE and ITEM)

2. For each relationship in the table on the right side, choose whether to "Remember" it or not based on the aforementioned rule.

3. The relationships that the user does not choose as "Remember" will be removed by default.

@Click on each Entity1 in the table to show the sentences where it is used. This can be helpful in identifying the appropriate relationship.

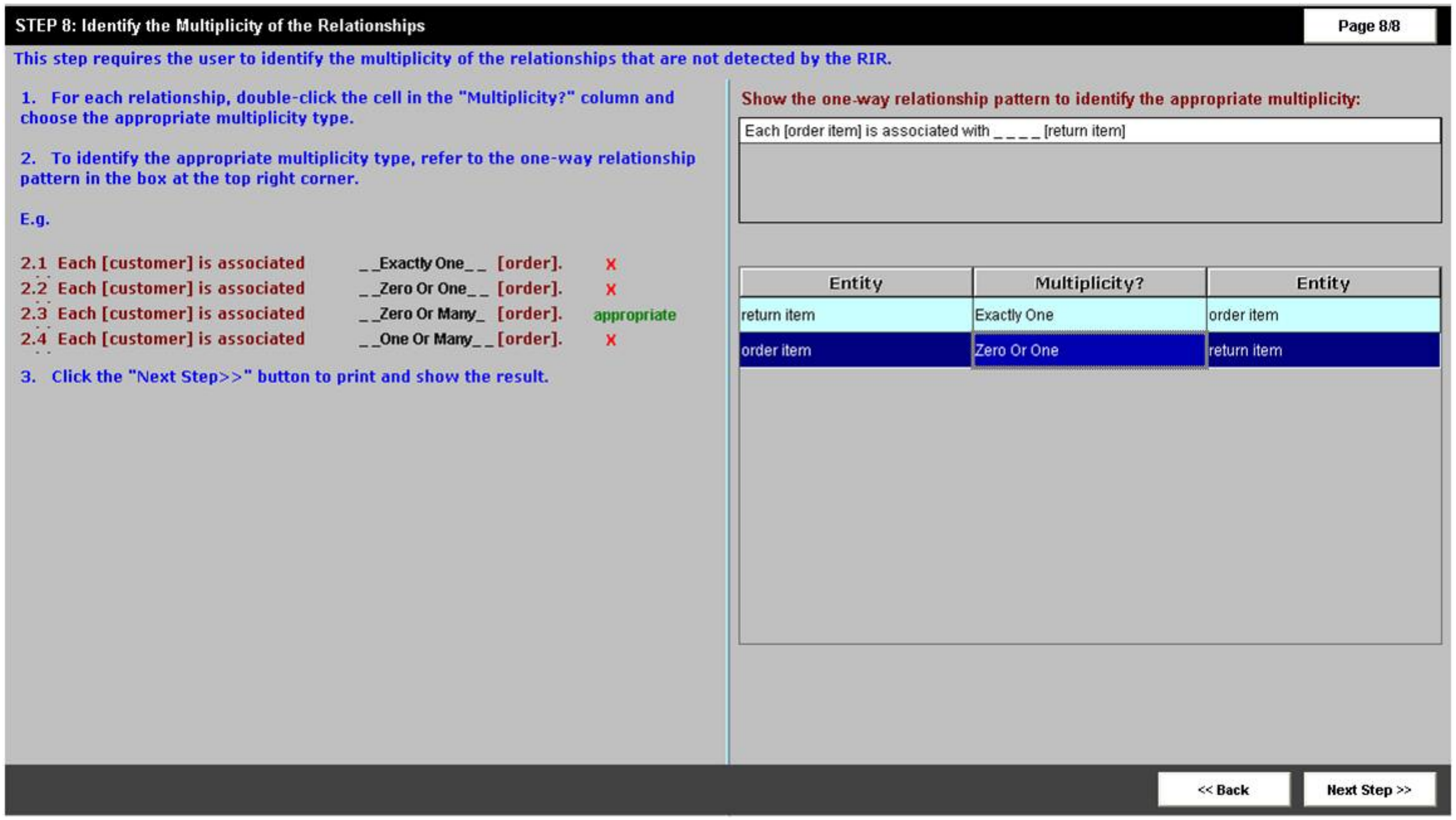
| Entity1 | Entity2 | Remember? |
|---|---|---|
| customer | login history | ☐ |
| return item | order item | ✓ |
| return item | price history | ☐ |
| return item | product | ☐ |
| return item | order | ☐ |
| return item | credit card | ☐ |
| return item | company | ☐ |
| return item | login history | ☐ |
| order item | price history | ☐ |
| order item | credit card | ☐ |
| order item | company | ☐ |

Show the sentence where an Entity1 is used:

1. For each return item, we keep track of return date, and total return price.

Figure A.8 A Screenshot of Step 7 in EIPW

Step 8: Assign the multiplicity to each relationship obtained from previous Step.



This step requires the user to identify the multiplicity of the relationships that are not detected by the RIR.

1. For each relationship, double-click the cell in the "Multiplicity?" column and choose the appropriate multiplicity type.

2. To identify the appropriate multiplicity type, refer to the one-way relationship pattern in the box at the top right corner.

E.g.

2.1 Each [customer] is associated __Exactly One__ [order]. x
2.2 Each [customer] is associated __Zero Or One__ [order]. x
2.3 Each [customer] is associated __Zero Or Many_ [order]. appropriate
2.4 Each [customer] is associated __One Or Many__[order]. x

3. Click the "Next Step>>" button to print and show the result.

Show the one-way relationship pattern to identify the appropriate multiplicity:

Each [order item] is associated with _ _ _ _ [return item]

| Entity | Multiplicity? | Entity |
|---|---|---|
| return item | Exactly One | order item |
| order item | Zero Or One | return item |

Figure A.9 A Screenshot of Step 8 in EIPW

Step 9: Show an output of EIPW.

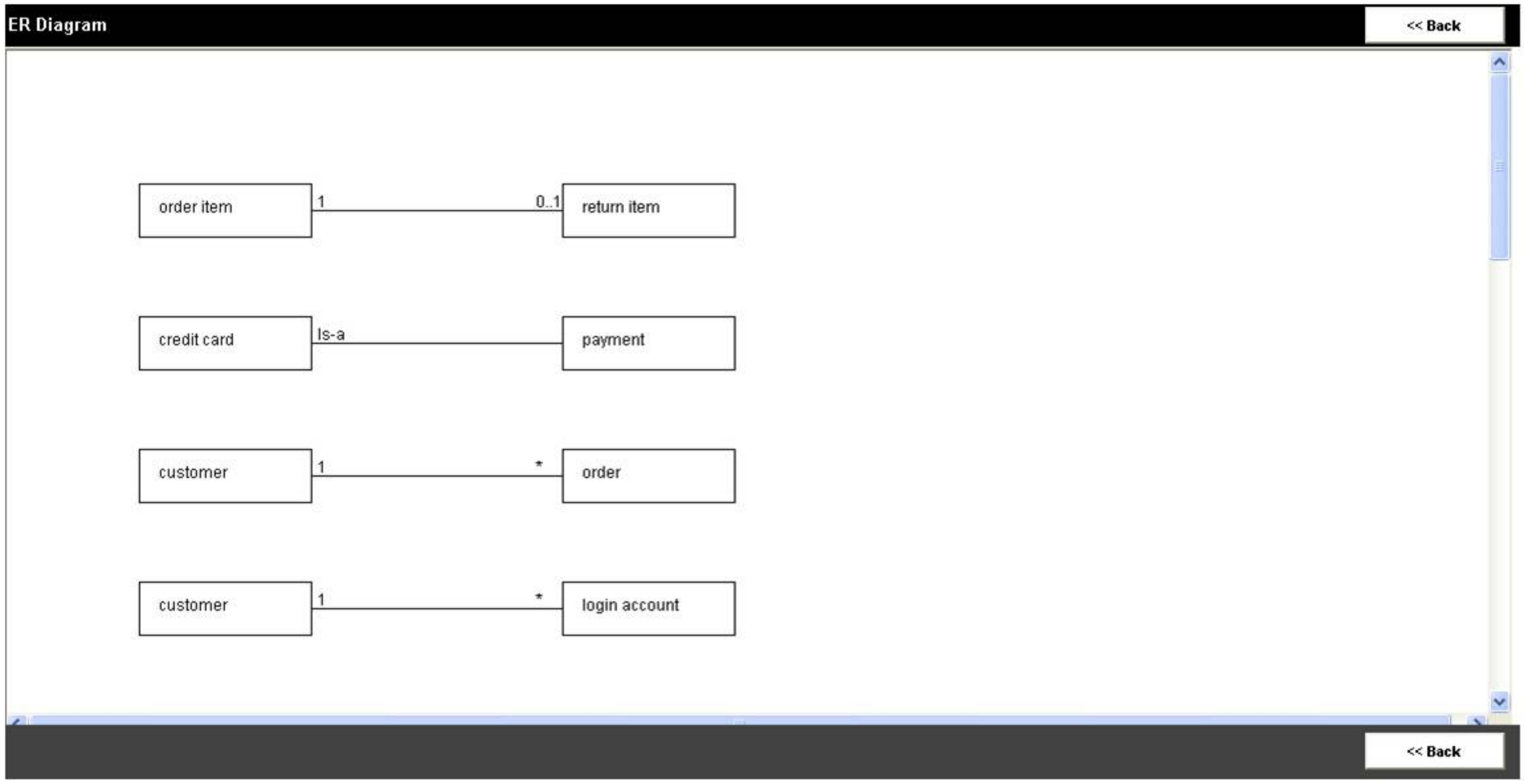
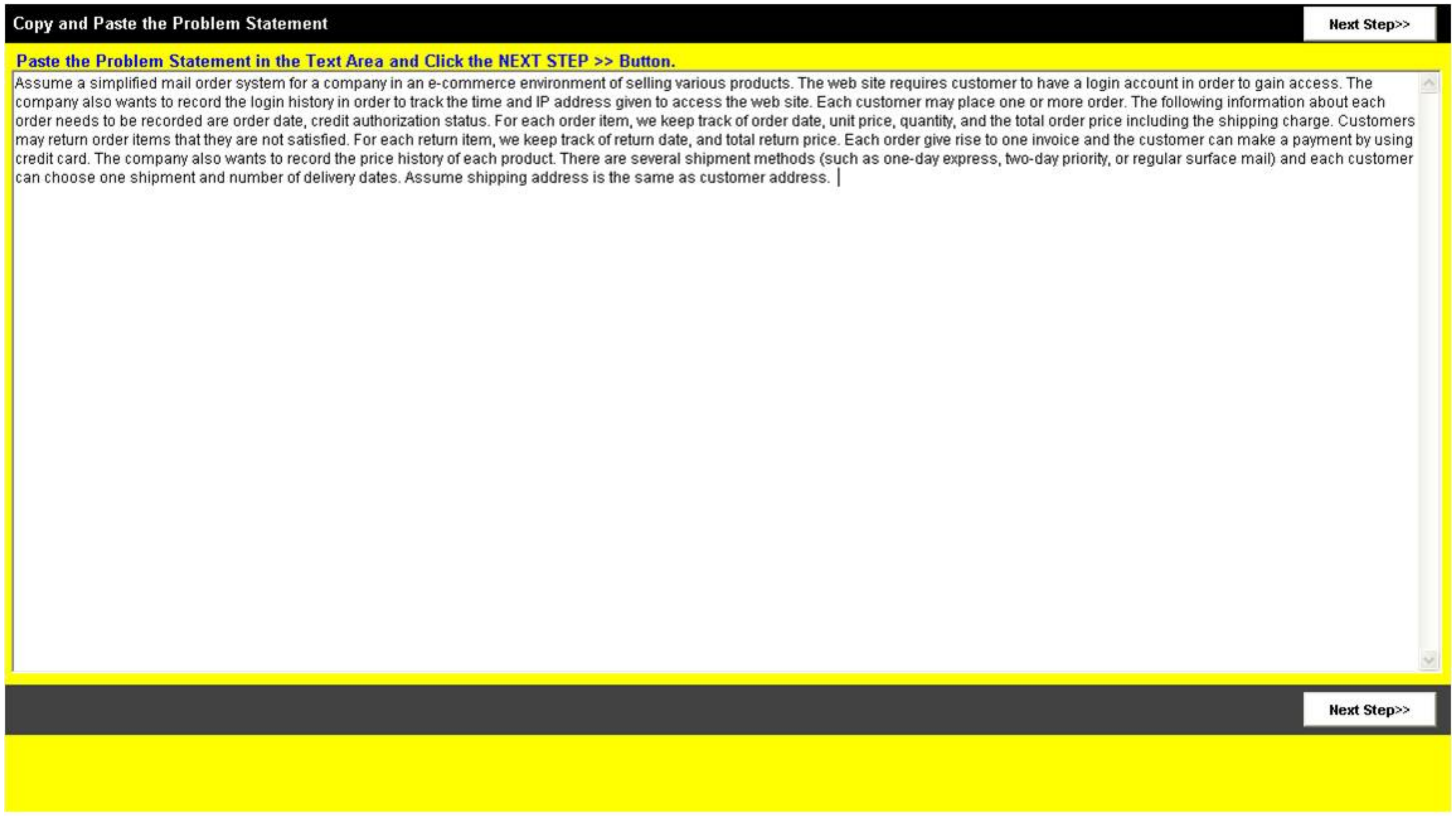


Figure A.10 An output of EIPW

## Appendix B: The user interfaces of HBT

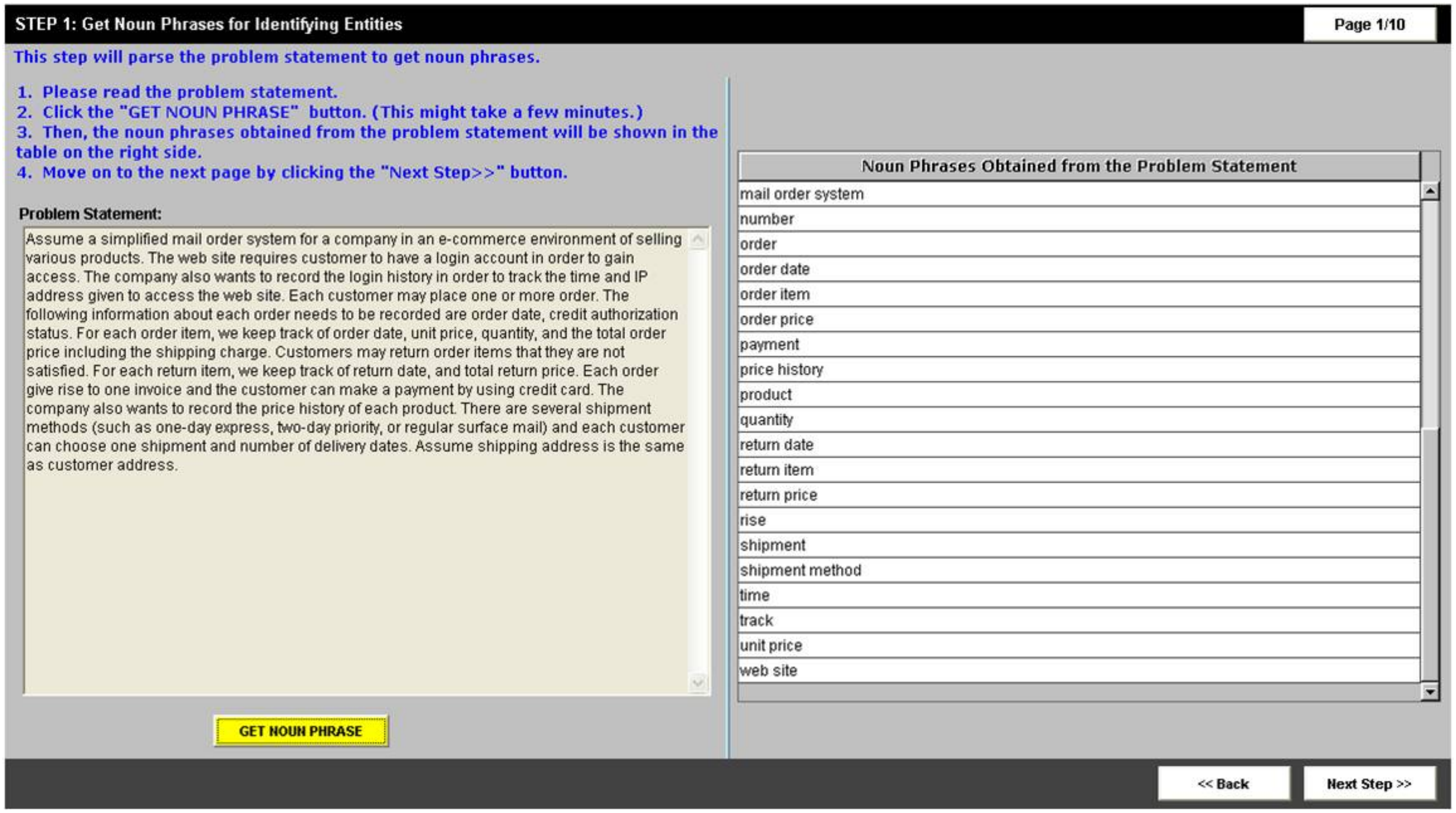
Initial Step: Begin with copy and paste a requirement specification in the text box.


Copy and Paste the Problem Statement
Next Step>>
Paste the Problem Statement in the Text Area and Click the NEXT STEP >> Button.
Next Step>>

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

Figure B.1 A Screenshot of Initial Step in HBT

Step 1: Get all the noun phrases from the requirement specification by applying POS.



This step will parse the problem statement to get noun phrases.

1. Please read the problem statement.
2. Click the "GET NOUN PHRASE" button. (This might take a few minutes.)
3. Then, the noun phrases obtained from the problem statement will be shown in the table on the right side.
4. Move on to the next page by clicking the "Next Step>>" button.

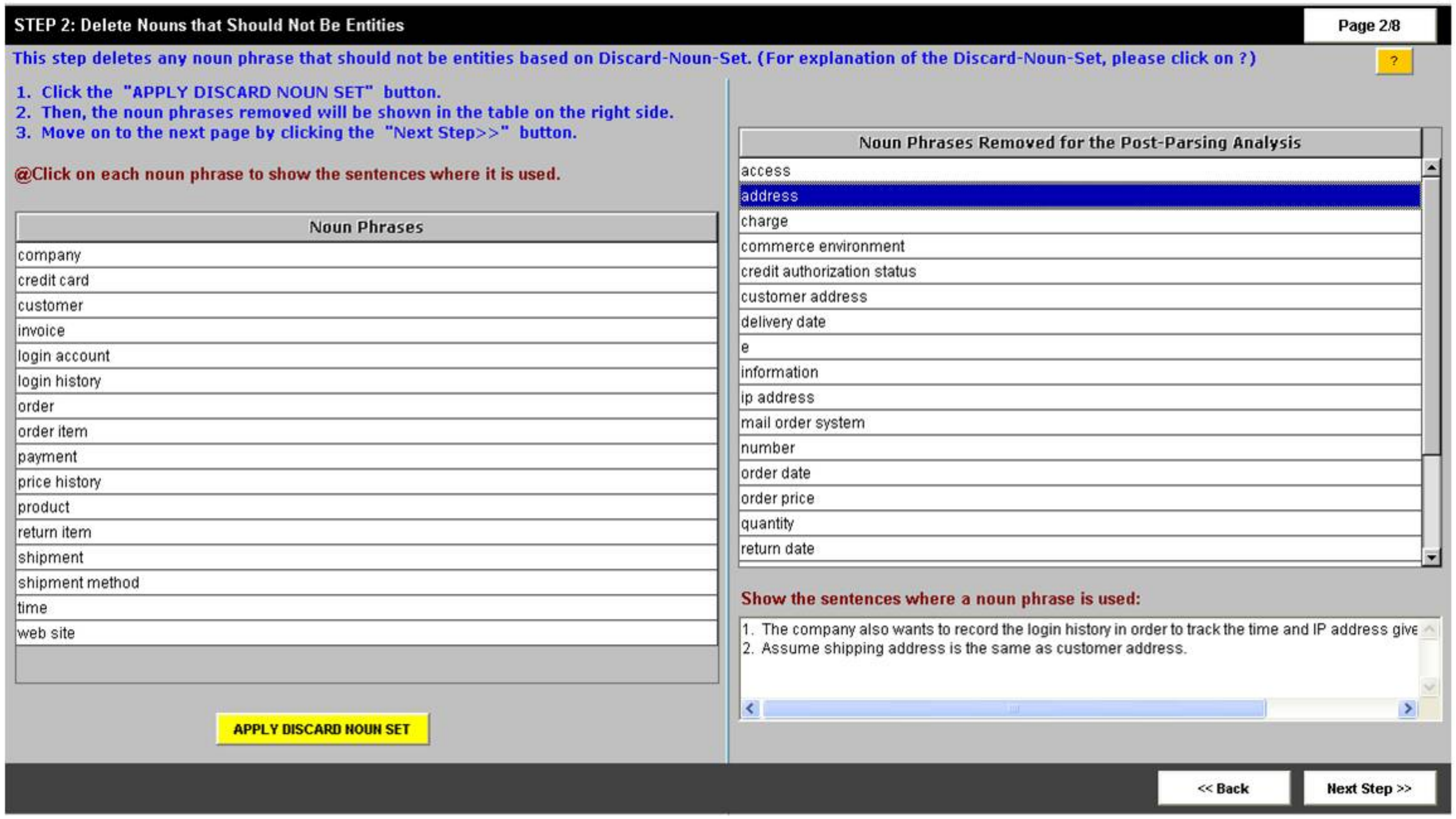
**Problem Statement:**

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

| Noun Phrases Obtained from the Problem Statement |
|---|
| mail order system |
| number |
| order |
| order date |
| order item |
| order price |
| payment |
| price history |
| product |
| quantity |
| return date |
| return item |
| return price |
| rise |
| shipment |
| shipment method |
| time |
| track |
| unit price |
| web site |

Figure B.2  A Screenshot of Step 1 in HBT

Step2: Test the discard noun set for facilitating the post-parsing analysis.



STEP 2: Delete Nouns that Should Not Be Entities

Page 2/8

This step deletes any noun phrase that should not be entities based on Discard-Noun-Set. (For explanation of the Discard-Noun-Set, please click on ?)

1. Click the "APPLY DISCARD NOUN SET" button.
2. Then, the noun phrases removed will be shown in the table on the right side.
3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase to show the sentences where it is used.

| Noun Phrases |
|---|
| company |
| credit card |
| customer |
| invoice |
| login account |
| login history |
| order |
| order item |
| payment |
| price history |
| product |
| return item |
| shipment |
| shipment method |
| time |
| web site |

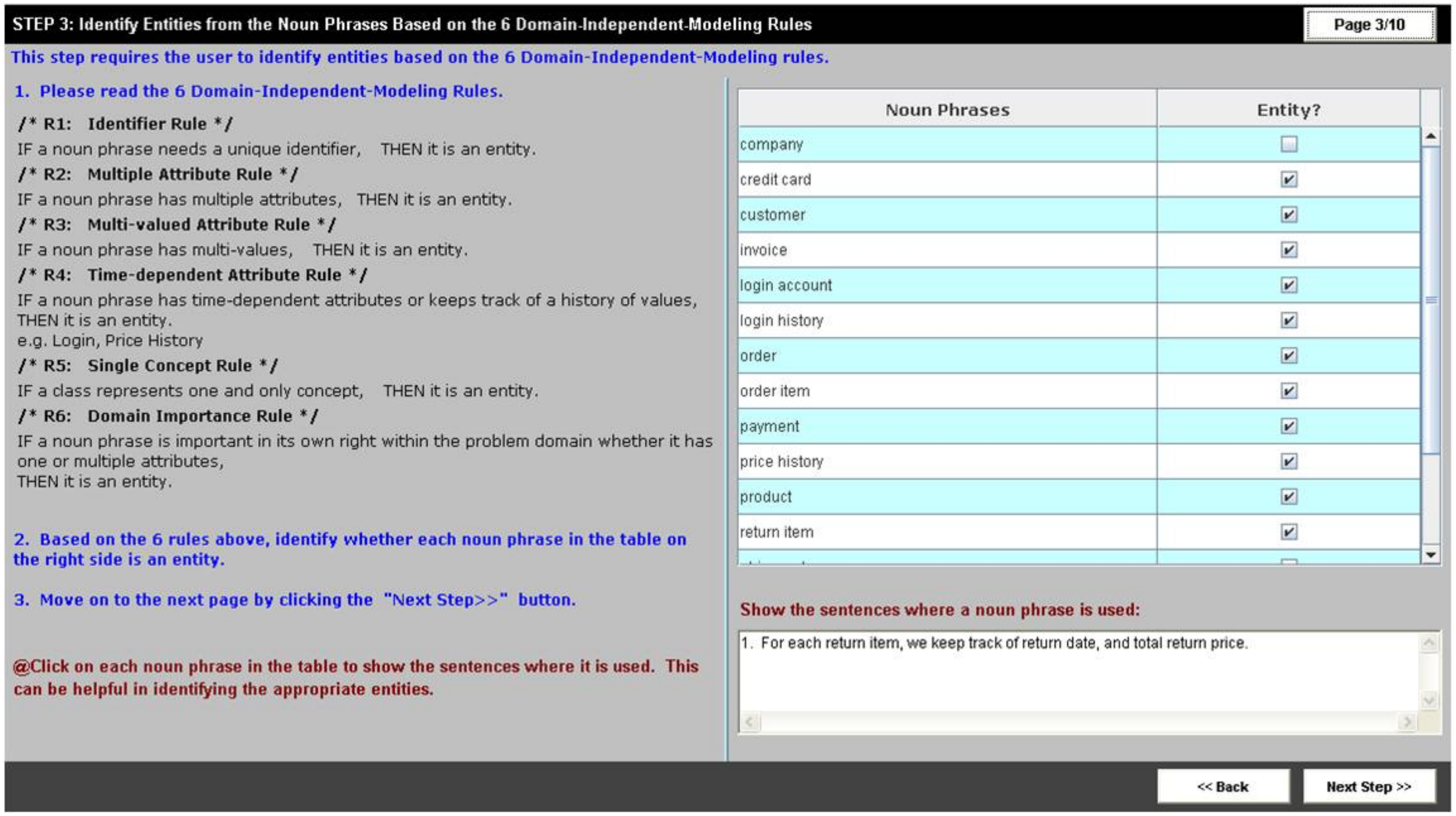
| Noun Phrases Removed for the Post-Parsing Analysis |
|---|
| access |
| address |
| charge |
| commerce environment |
| credit authorization status |
| customer address |
| delivery date |
| e |
| information |
| ip address |
| mail order system |
| number |
| order date |
| order price |
| quantity |
| return date |

Show the sentences where a noun phrase is used:

1. The company also wants to record the login history in order to track the time and IP address give
2. Assume shipping address is the same as customer address.

Figure B.3 A Screenshot of Step 2 in HBT

Step 3: Identify the entities from noun phrases based on six domain independent modeling rules.



**STEP 3: Identify Entities from the Noun Phrases Based on the 6 Domain-Independent-Modeling Rules**

Page 3/10

This step requires the user to identify entities based on the 6 Domain-Independent-Modeling rules.

1. Please read the 6 Domain-Independent-Modeling Rules.

**/* R1: Identifier Rule */**
IF a noun phrase needs a unique identifier, THEN it is an entity.

**/* R2: Multiple Attribute Rule */**
IF a noun phrase has multiple attributes, THEN it is an entity.

**/* R3: Multi-valued Attribute Rule */**
IF a noun phrase has multi-values, THEN it is an entity.

**/* R4: Time-dependent Attribute Rule */**
IF a noun phrase has time-dependent attributes or keeps track of a history of values, THEN it is an entity.
e.g. Login, Price History

**/* R5: Single Concept Rule */**
IF a class represents one and only concept, THEN it is an entity.

**/* R6: Domain Importance Rule */**
IF a noun phrase is important in its own right within the problem domain whether it has one or multiple attributes,
THEN it is an entity.

2. Based on the 6 rules above, identify whether each noun phrase in the table on the right side is an entity.

3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each noun phrase in the table to show the sentences where it is used. This can be helpful in identifying the appropriate entities.

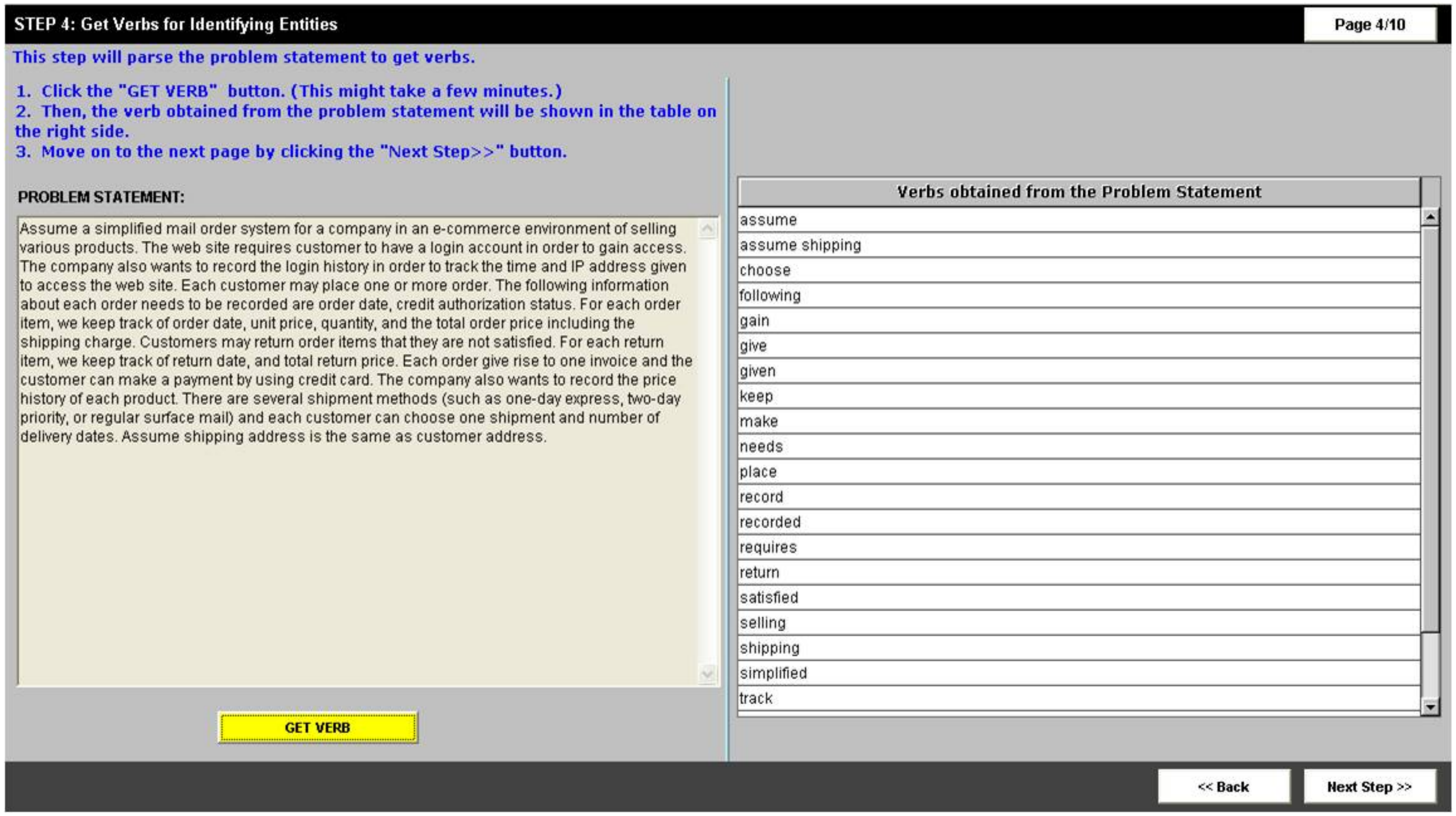
| Noun Phrases | Entity? |
|---|---|
| company | ☐ |
| credit card | ✓ |
| customer | ✓ |
| invoice | ✓ |
| login account | ✓ |
| login history | ✓ |
| order | ✓ |
| order item | ✓ |
| payment | ✓ |
| price history | ✓ |
| product | ✓ |
| return item | ✓ |

Show the sentences where a noun phrase is used:

1. For each return item, we keep track of return date, and total return price.

Figure B.4 A Screenshot of Step 3 in HBT

Step 4: Get all the verb phrases from the requirement specification by applying POS.



This step will parse the problem statement to get verbs.

1. Click the "GET VERB" button. (This might take a few minutes.)
2. Then, the verb obtained from the problem statement will be shown in the table on the right side.
3. Move on to the next page by clicking the "Next Step>>" button.

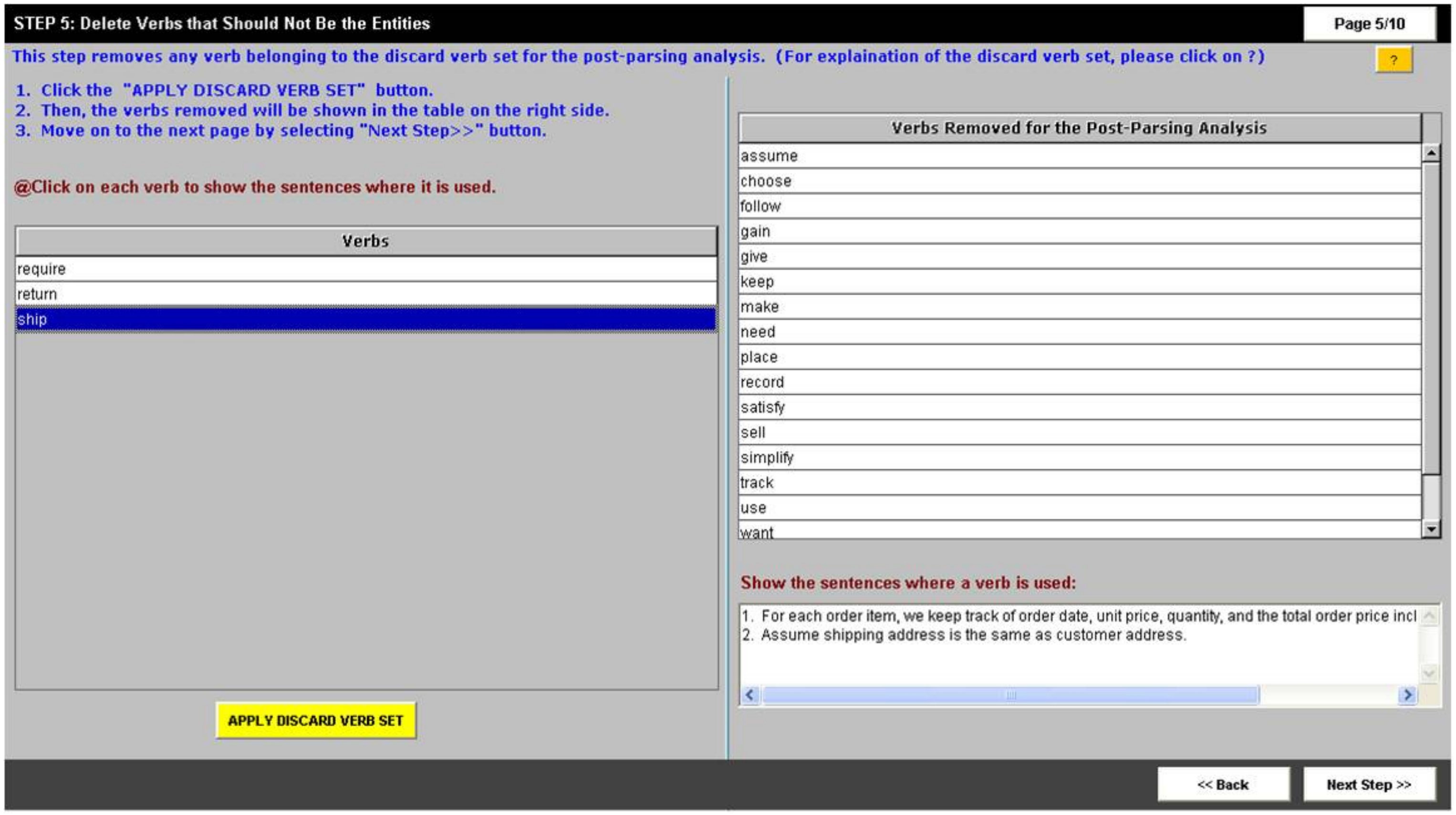
PROBLEM STATEMENT:

Assume a simplified mail order system for a company in an e-commerce environment of selling various products. The web site requires customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and number of delivery dates. Assume shipping address is the same as customer address.

| Verbs obtained from the Problem Statement |
|---|
| assume |
| assume shipping |
| choose |
| following |
| gain |
| give |
| given |
| keep |
| make |
| needs |
| place |
| record |
| recorded |
| requires |
| return |
| satisfied |
| selling |
| shipping |
| simplified |
| track |

Figure B.5 A Screenshot of Step 4 in HBT

Step 5: Test the discard verb set for facilitating the post-parsing analysis.



STEP 5: Delete Verbs that Should Not Be the Entities

Page 5/10

This step removes any verb belonging to the discard verb set for the post-parsing analysis. (For explaination of the discard verb set, please click on ?)

1. Click the "APPLY DISCARD VERB SET" button.
2. Then, the verbs removed will be shown in the table on the right side.
3. Move on to the next page by selecting "Next Step>>" button.

@Click on each verb to show the sentences where it is used.

| Verbs |
|---|
| require |
| return |
| ship |

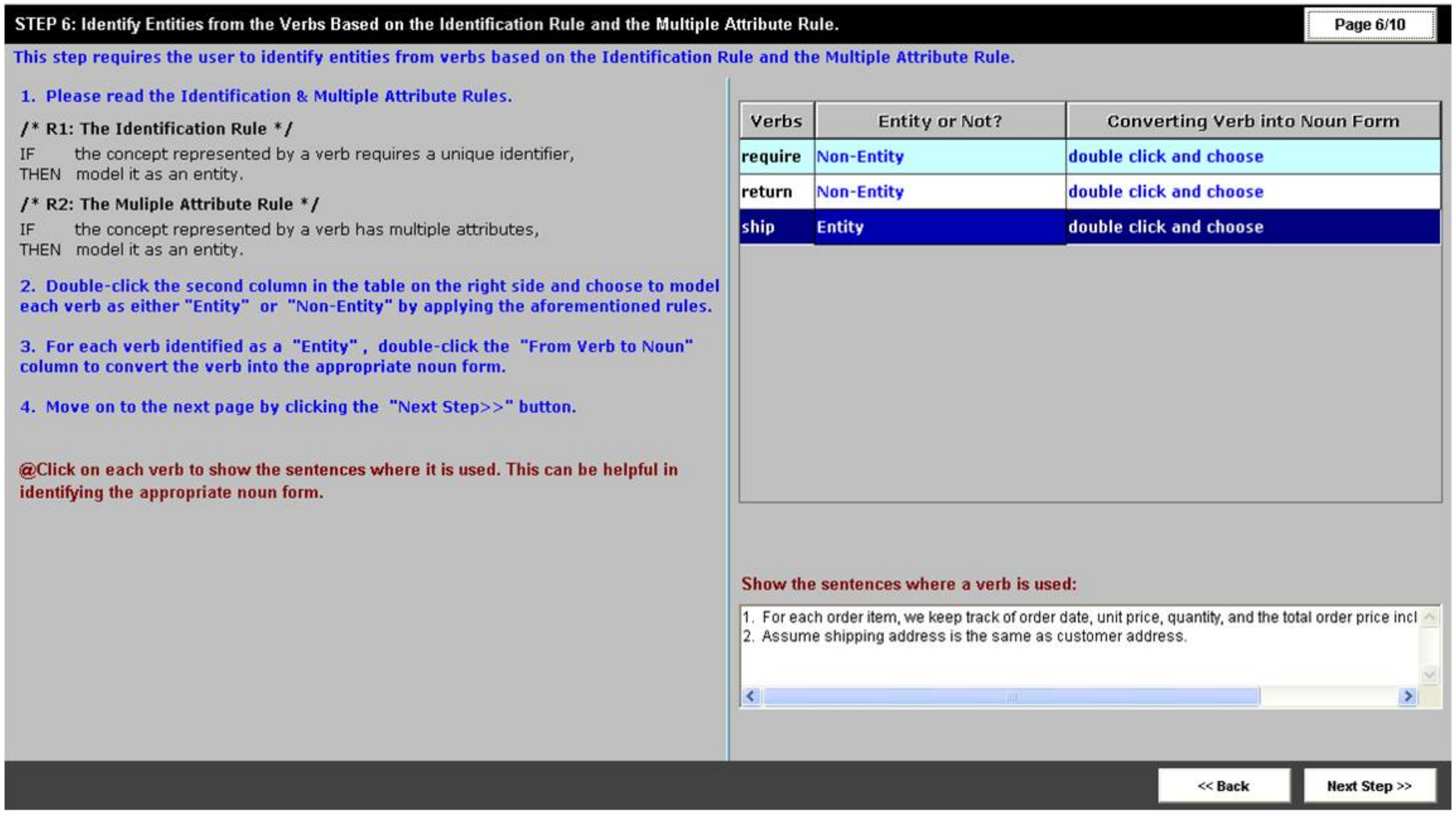
| Verbs Removed for the Post-Parsing Analysis |
|---|
| assume |
| choose |
| follow |
| gain |
| give |
| keep |
| make |
| need |
| place |
| record |
| satisfy |
| sell |
| simplify |
| track |
| use |
| want |

Show the sentences where a verb is used:

1. For each order item, we keep track of order date, unit price, quantity, and the total order price incl
2. Assume shipping address is the same as customer address.

Figure B.6 A Screenshot of Step 5 in HBT

Step 6: Identify the entities from verb phrases based on modeling rules.



STEP 6: Identify Entities from the Verbs Based on the Identification Rule and the Multiple Attribute Rule.

Page 6/10

This step requires the user to identify entities from verbs based on the Identification Rule and the Multiple Attribute Rule.

1. Please read the Identification & Multiple Attribute Rules.

/* R1: The Identification Rule */

IF the concept represented by a verb requires a unique identifier,
THEN model it as an entity.

/* R2: The Muliple Attribute Rule */

IF the concept represented by a verb has multiple attributes,
THEN model it as an entity.

2. Double-click the second column in the table on the right side and choose to model each verb as either "Entity" or "Non-Entity" by applying the aforementioned rules.

3. For each verb identified as a "Entity", double-click the "From Verb to Noun" column to convert the verb into the appropriate noun form.

4. Move on to the next page by clicking the "Next Step>>" button.

@Click on each verb to show the sentences where it is used. This can be helpful in identifying the appropriate noun form.

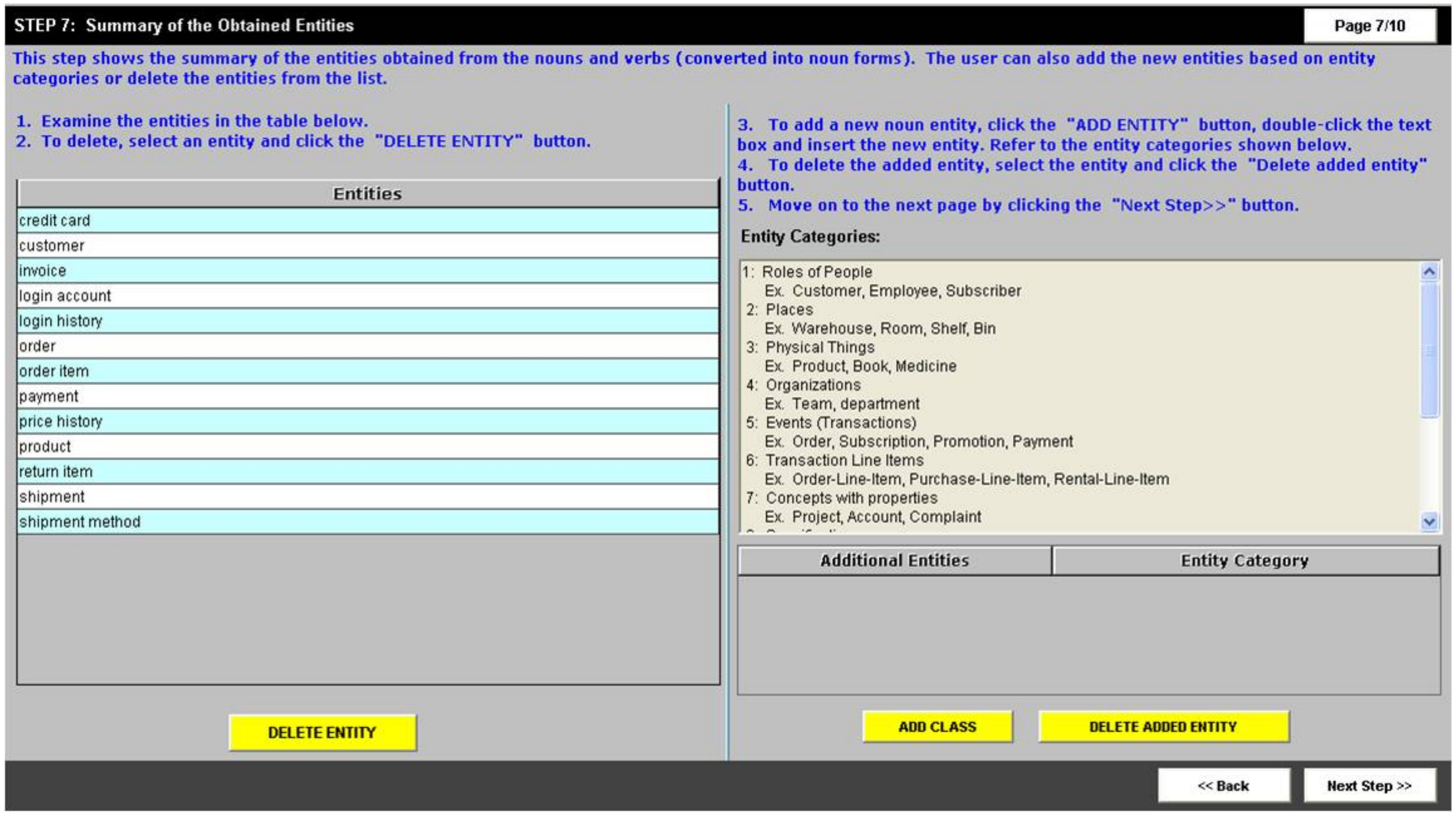
| Verbs | Entity or Not? | Converting Verb into Noun Form |
|---|---|---|
| require | Non-Entity | double click and choose |
| return | Non-Entity | double click and choose |
| ship | Entity | double click and choose |

Show the sentences where a verb is used:

1. For each order item, we keep track of order date, unit price, quantity, and the total order price incl
2. Assume shipping address is the same as customer address.

Figure B.7 A Screenshot of Step 6 in HBT

Step 7: Identify the entities based on entity categories.



This step shows the summary of the entities obtained from the nouns and verbs (converted into noun forms). The user can also add the new entities based on entity categories or delete the entities from the list.

1. Examine the entities in the table below.
2. To delete, select an entity and click the "DELETE ENTITY" button.

| Entities |
|---|
| credit card |
| customer |
| invoice |
| login account |
| login history |
| order |
| order item |
| payment |
| price history |
| product |
| return item |
| shipment |
| shipment method |

3. To add a new noun entity, click the "ADD ENTITY" button, double-click the text box and insert the new entity. Refer to the entity categories shown below.
4. To delete the added entity, select the entity and click the "Delete added entity" button.
5. Move on to the next page by clicking the "Next Step>>" button.

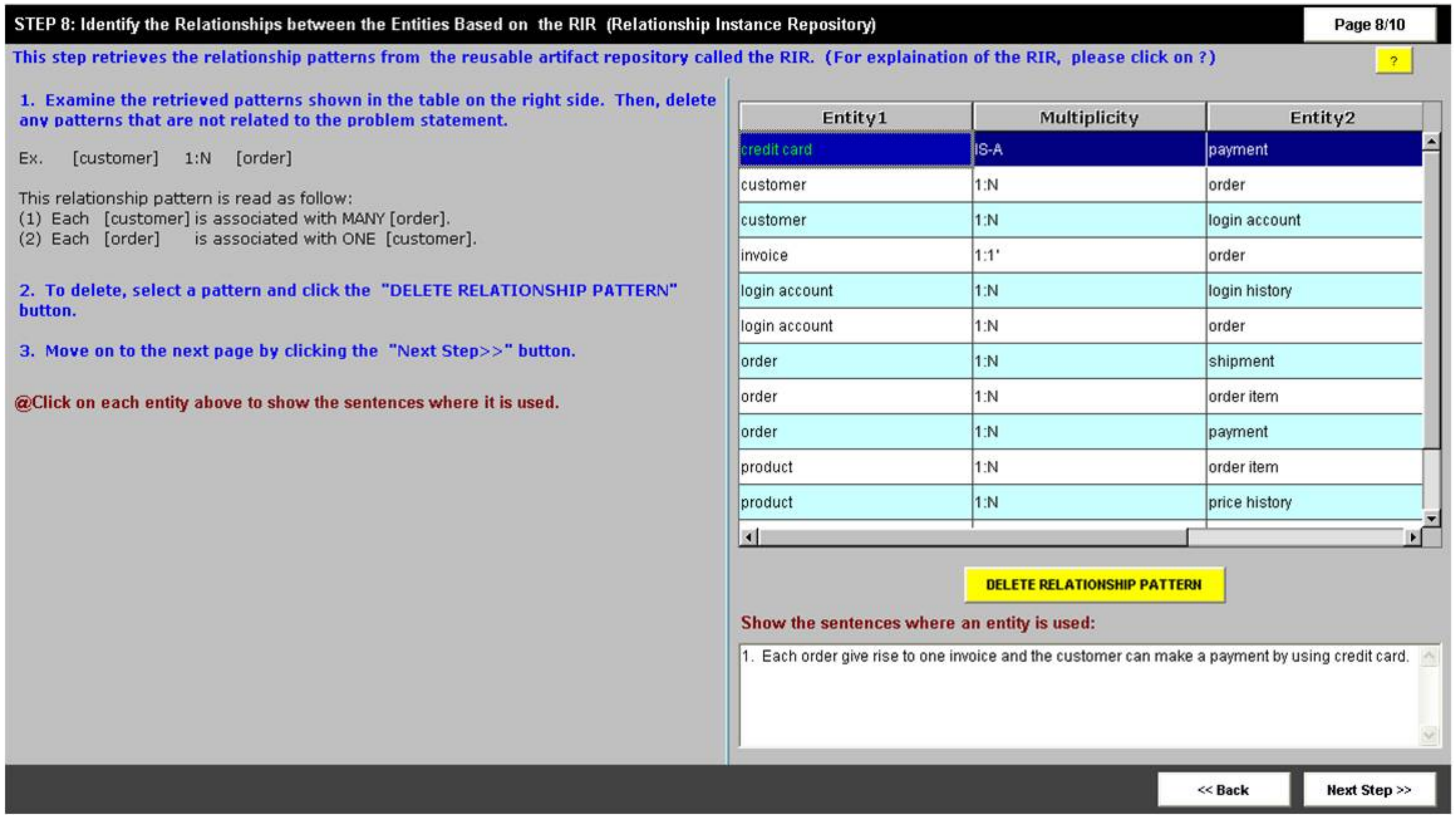
Entity Categories:

1: Roles of People
Ex. Customer, Employee, Subscriber
2: Places
Ex. Warehouse, Room, Shelf, Bin
3: Physical Things
Ex. Product, Book, Medicine
4: Organizations
Ex. Team, department
5: Events (Transactions)
Ex. Order, Subscription, Promotion, Payment
6: Transaction Line Items
Ex. Order-Line-Item, Purchase-Line-Item, Rental-Line-Item
7: Concepts with properties
Ex. Project, Account, Complaint

| Additional Entities | Entity Category |
|---|---|

Figure B.8 A Screenshot of Step 7 in HBT

Step 8: identify relationships based on RIR (Relationship Instance Repository).



STEP 8: Identify the Relationships between the Entities Based on the RIR (Relationship Instance Repository)

Page 8/10

This step retrieves the relationship patterns from the reusable artifact repository called the RIR. (For explaination of the RIR, please click on ?)

1. Examine the retrieved patterns shown in the table on the right side. Then, delete any patterns that are not related to the problem statement.

Ex. [customer] 1:N [order]

This relationship pattern is read as follow:
(1) Each [customer] is associated with MANY [order].
(2) Each [order] is associated with ONE [customer].

2. To delete, select a pattern and click the "DELETE RELATIONSHIP PATTERN" button.

3. Move on to the next page by clicking the "Next Step>>" button.

@Click on each entity above to show the sentences where it is used.

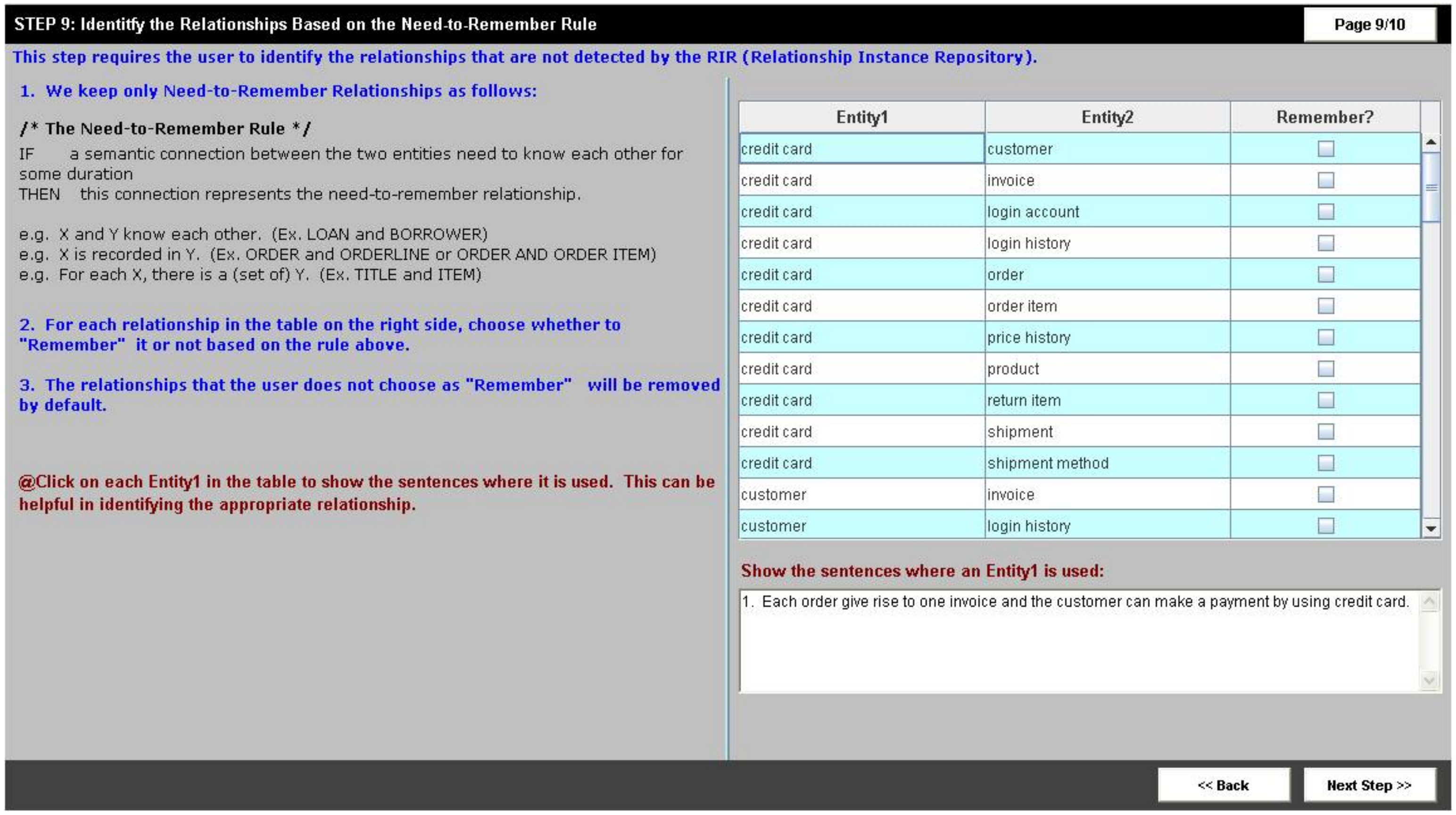
| Entity1 | Multiplicity | Entity2 |
|---|---|---|
| credit card | IS-A | payment |
| customer | 1:N | order |
| customer | 1:N | login account |
| invoice | 1:1' | order |
| login account | 1:N | login history |
| login account | 1:N | order |
| order | 1:N | shipment |
| order | 1:N | order item |
| order | 1:N | payment |
| product | 1:N | order item |
| product | 1:N | price history |

Show the sentences where an entity is used:

1. Each order give rise to one invoice and the customer can make a payment by using credit card.

Figure B.9 A Screenshot of Step 8 in HBT

Step 9: Identify relationships, which are not detected by RIR, by applying a modeling rule.


STEP 9: Identitfy the Relationships Based on the Need-to-Remember Rule

Page 9/10

This step requires the user to identify the relationships that are not detected by the RIR (Relationship Instance Repository).

1. We keep only Need-to-Remember Relationships as follows:

/* The Need-to-Remember Rule */
IF a semantic connection between the two entities need to know each other for some duration
THEN this connection represents the need-to-remember relationship.

e.g. X and Y know each other. (Ex. LOAN and BORROWER)
e.g. X is recorded in Y. (Ex. ORDER and ORDERLINE or ORDER AND ORDER ITEM)
e.g. For each X, there is a (set of) Y. (Ex. TITLE and ITEM)

2. For each relationship in the table on the right side, choose whether to "Remember" it or not based on the rule above.

3. The relationships that the user does not choose as "Remember" will be removed by default.

@Click on each Entity1 in the table to show the sentences where it is used. This can be helpful in identifying the appropriate relationship.

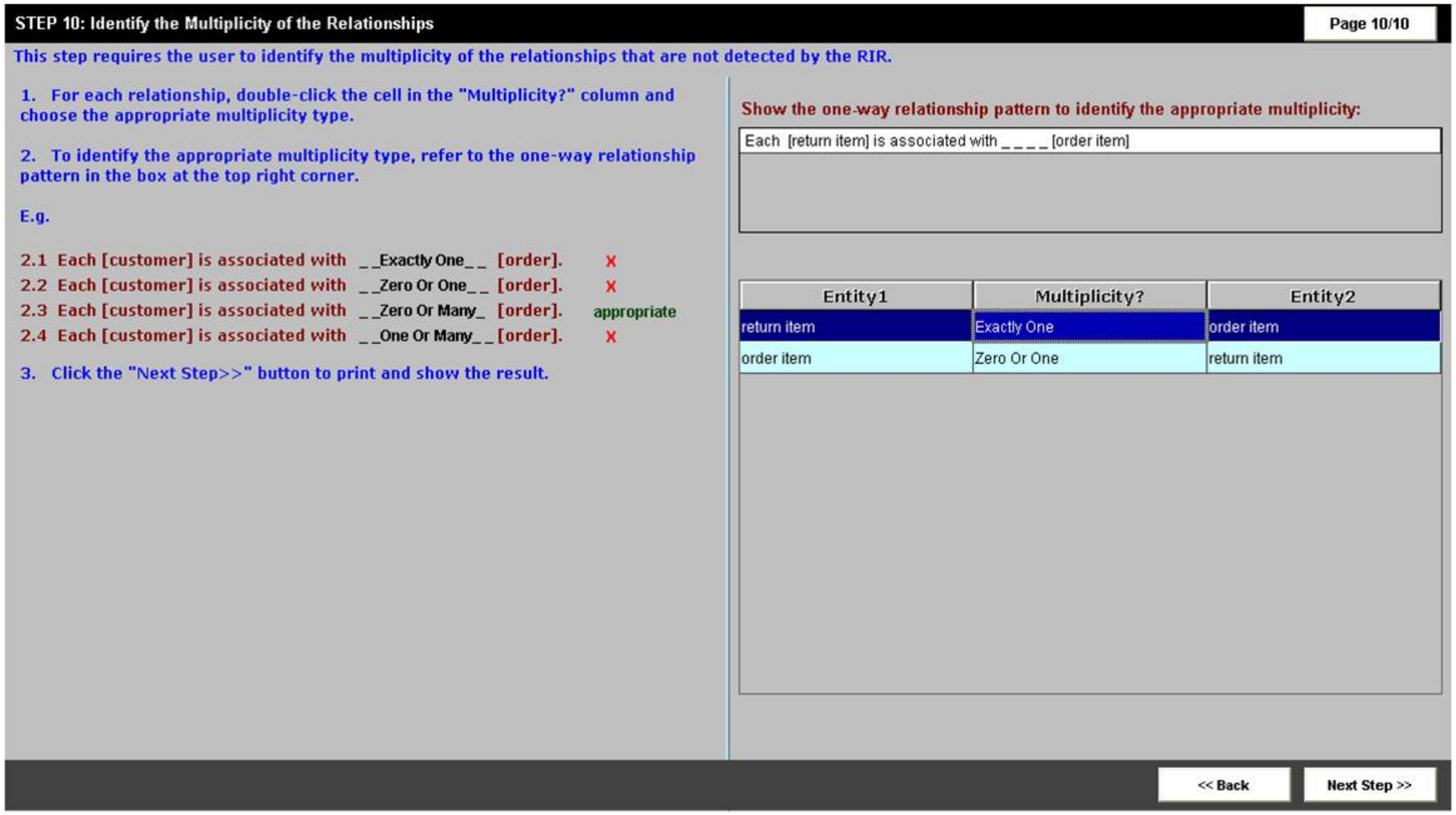
| Entity1 | Entity2 | Remember? |
|---|---|---|
| credit card | customer | ☐ |
| credit card | invoice | ☐ |
| credit card | login account | ☐ |
| credit card | login history | ☐ |
| credit card | order | ☐ |
| credit card | order item | ☐ |
| credit card | price history | ☐ |
| credit card | product | ☐ |
| credit card | return item | ☐ |
| credit card | shipment | ☐ |
| credit card | shipment method | ☐ |
| customer | invoice | ☐ |
| customer | login history | ☐ |

Show the sentences where an Entity1 is used:

1. Each order give rise to one invoice and the customer can make a payment by using credit card.

Figure B. 10 A Screenshot of Step 9 in HBT

Step 10: Assign the multiplicity to each relationship obtained from preveios Step.



STEP 10: Identify the Multiplicity of the Relationships

Page 10/10

This step requires the user to identify the multiplicity of the relationships that are not detected by the RIR.

1. For each relationship, double-click the cell in the "Multiplicity?" column and choose the appropriate multiplicity type.

2. To identify the appropriate multiplicity type, refer to the one-way relationship pattern in the box at the top right corner.

E.g.

2.1 Each [customer] is associated with __Exactly One__ [order]. X
2.2 Each [customer] is associated with __Zero Or One__ [order]. X
2.3 Each [customer] is associated with __Zero Or Many_ [order]. appropriate
2.4 Each [customer] is associated with __One Or Many__ [order]. X

3. Click the "Next Step>>" button to print and show the result.

Show the one-way relationship pattern to identify the appropriate multiplicity:

Each [return item] is associated with ____ [order item]

| Entity1 | Multiplicity? | Entity2 |
|---|---|---|
| return item | Exactly One | order item |
| order item | Zero Or One | return item |

Figure B.11 A Screenshot of Step 10 in HBT

Step 11: Show an output of HBT

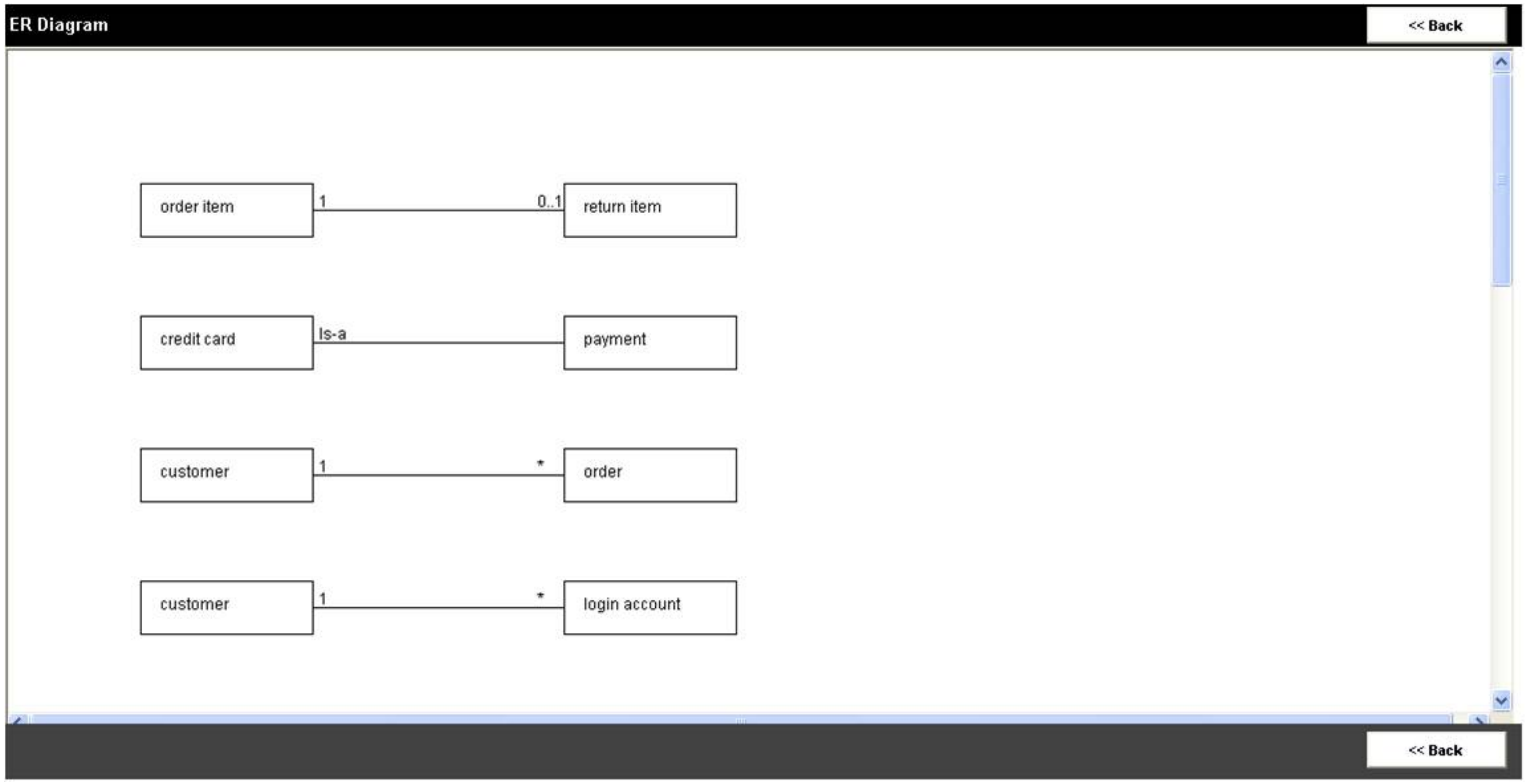


Figure B.12 An output of HB

# Appendix C: Experimental Problem Statements

Problem 1 (Moderate Task Size)

Assume a simplified mail order system for a company in an e-commerce environment of selling various products for children. The web site requires a customer to have a login account in order to gain access. The company also wants to record the login history in order to track the time and IP address given to access the web site. Each customer may place one or more order. The following information about each order needs to be recorded are order date, credit authorization status. For each order item, we keep track of order date, unit price, quantity, and the total order price including the shipping charge. Customers may return order items that they are not satisfied. For each return item, we keep track of return date, and total return price. Each order give rise to one invoice and the customer can make a payment by using credit card (CC), where we keep track of credit card types (Visa, MasterCard, AMEX), CC number, name on the CC, and expiration date. The company also wants to record the price history of each product. There are several shipment methods (such as one-day express, two-day priority, or regular surface mail) and each customer can choose one shipment and numbers of delivery dates. Assume the shipping charge is a straightforward percentage of the total order price plus extra charge depending on the shipping method. Shipping address is the same as the customer address.

Problem 2 (Medium Task Size)

The company wants to create a database to keep track of all employees and projects assigned. Projects are distinguished by project numbers, the customers to which the project belongs, a project start date, a project end date, and estimated. In the project plan, the manager of the project must determine the tasks that will be performed to take the project from beginning to the end. Each task has a task ID, a task description, the task's starting and ending dates, and the number of employees required completing the task. Employees are assigned to specific tasks scheduled by the manager. The hour's employees working are kept in a work log entry containing a record of the actual hours worked by an employee on a given assignment. The work log is a weekly form that the employee fills out at the end of each month. The work log also contains the date (the last workday of the month), assignment ID, the total hours worked up to the end of that month, and the number of the bill to which the work-log entry is charged. Obviously, each work log entry can be related to only one bill. The company has pooled all of its employees by region, and from this pool, employees are assigned to a specific task.

Problem 3 (Moderate Task Size)

Temporary Employment Corporation (TEC) places the temporary workers in companies.TEC has a file of candidates who are willing to work. If the candidate has worked before, that candidate has a specific job history (Naturally, no job history exists if the candidate has never worked). Each time the candidate works, one additional job history record is created. Each candidate has earned several qualifications. TEC offers courses to help candidates improve their qualifications. Every course develops one specific qualification. Some qualifications have multiple courses that develop that qualification. Some courses require specific qualifications as prerequisites. A course can have several prerequisites. Courses are taught during the training sessions with specialists in the field. A training session is the presentation of a single course and is scheduled in a particular room at a specific time slot. Candidates can register and pay a fee to attend a training session. TEC also has a list of companies that request temporaries. Each time a company requests a temporary employee, TEC makes an entry in the Open Position folder. That folder contains an opening number, a company name, required qualifications, a starting date, an anticipated ending date, and hourly pay. Each opening position requires only one specific or main qualification. When a candidate matches the qualification, the placement is assigned, and an entry is made in the Placement Record folder. That folder contains a placement date, a placement number, the total hours worked, etc. In addition, an entry is made in the job history for the candidate. A placement can be filled by many candidates, and a candidate can fill many placements.

<u>Problem 4 (Medium Task Size)</u>

A company wishes to create a database to control its inventory, which consists of many products divided into a number of product categories (i.e. clothing, shoes, bags, and accessories). The purchase department makes a purchase order when a product has to be reordered from the suppliers. The name of the employee who processes the purchase order is record. Each purchase line item records the purchase ID, product received, cost, and any wastage. The reorder guideline provides information on how to best reorder products. The same products are stored in the same warehouse, where the company have several different warehouses located in different cities. The different tax rates are used in different states. The sales tax is not applied to every purchase order, but only occurs in those states in which the merchant is required by the state to collect taxes on products sold in the state. Normally, the sales tax determination could be based on the location of the supplier.

# Appendix D: Pre-Experiment Questionnaires

This questionnaire is part of an analysis of a research work in the area of conceptual modeling. Your answers will be kept confidential. Thank you for your cooperation.

1. Gender (circle your selection): Male / Female
2. Age: <20 20-29 30-39 40-49 50-59
3. Year or Degree Completed: Fresh. Soph. Jun. Sen. Grad.
4. In order to participate in this study, you must have some experiences with Entity-Relationship (ER) models for database design. Please indicate the type of experience you have with the ER modeling.
   - ☐ Developing ER models for database design classes some years ago.
   - ☐ Developing small ER models for class assignments or class project recently.
   - ☐ Extensive using ER models for database design.
   - ☐ Expert knowledge of using ER models for database design.
5. Level of difficulty in database topic:

| | Very Difficult | | | Very Easy |
|---|---|---|---|---|
| Entity-Relationship Modeling | ☐ | ☐ | ☐ | ☐ |
| Normalization | ☐ | ☐ | ☐ | ☐ |
| The Relational Model | ☐ | ☐ | ☐ | ☐ |

| | | | | |
|---|---|---|---|---|
| SQL | ☐ | ☐ | ☐ | ☐ |

6. Do you understand the concepts about entity, relationship, and attribute in ER model?

| | Very clear | | | very unclear |
|---|---|---|---|---|
| <u>Entity</u> | ☐ | ☐ | ☐ | ☐ |
| <u>Relationship</u> | ☐ | ☐ | ☐ | ☐ |
| <u>Attribute</u> | ☐ | ☐ | ☐ | ☐ |

7. How do you determine entities and relationships in ER modeling?

   ☐ Scanning the sentences in problem statement for nouns and verbs.

   ☐ Visualizing the scenario

   ☐ Guessing

   ☐ Others   Please specify: ___________________________

8. Have you created the ER model based on a given problem statement?

| Every time | | | Never |
|---|---|---|---|
| ☐ | ☐ | ☐ | ☐ |

9. Have you used any diagramming tools to help you in creating an ER model?

   ☐ Yes   If yes, please name: __________________________________

   ☐ No

10. Do the diagramming tools help in improving your skills in the ER modeling?

- ☐ Yes
- ☐ No

# Appendix E: Pre-Experiment Questionnaires

Please answer the questions about the experiment you just finished. Your answers will be kept confidential.

11. How difficult is the problem statements?

    Please rate in the scale below

| | Very difficult | | | very easy |
|---|---|---|---|---|
| Problem 1 | ☐ | ☐ | ☐ | ☐ |
| Problem 2 | ☐ | ☐ | ☐ | ☐ |
| Problem 3 | ☐ | ☐ | ☐ | ☐ |
| Problem 4 | ☐ | ☐ | ☐ | ☐ |

12. How confident do you feel about modeling the ERD (Entity-Relationship Diagram) to the problem statements?

    - ☐ Extremely confident
    - ☐ Somewhat confident
    - ☐ Neither confident or distrustful
    - ☐ Somewhat not confident
    - ☐ Not confident

13. In your opinion, rate the overall helpfulness of our modeling tool for developing ER models.

    - ☐ Very helpful
    - ☐ Somewhat helpful
    - ☐ A little helpful
    - ☐ Not at all helpful

14. In your opinion, rate the impact of our tool on how easy to use and user-friendly.

    - ☐ Not at all
    - ☐ Not very easy to use
    - ☐ Somewhat easy to use and user friendly
    - ☐ Very easy to use and user friendly

15. What are the main advantages of using our tool?

16. Identify areas where you believe that the conceptual data modeling tools could be improved. Please explain your improvement.

# Curriculum Vitae

The author was born in Bangkok, Thailand on September 7, 1974. She attended Chulalongkorn University from 1991 to 1995, and graduated with a Bachelor of Science. Later, she attended Rochester Institute of Technology from 1997 to 1999, and graduated with a Master of Science.

She came to Drexel University in the Fall of 2005 and began her Ph.D program in the College of Information Science and Technology, with a scholarship from the Royal Thai government. She pursued her research in automated conceptual data modeling under the supervision of Professor Il-Yeol Song.