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

Ornsiri Thonggoom, Il-Yeol Song, and Yuan An

The iSchool at Drexel University, Philadelphia, PA USA
Ot62@drexel.edu, songiy@drexel.edu, yuan.an@drexel.edu

Abstract. Data modelers frequently lack experience and have incomplete knowledge about the application being designed. To address this issue, we propose new types of reusable artifacts called Entity Instance Repository (EIR) and Relationship Instance Repository (RIR), which contain ER modeling patterns from prior designs and serve as knowledge-based repositories for conceptual modeling. We explore the development of automated data modeling tools with EIR and RIR. We also select six data modeling rules used for identification of entities in one of the tools. Two tools 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 modelers' 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 tools are effective in creating quality data models. In order to evaluate the effectiveness of the tools, empirical testing was conducted on tasks of different sizes. The empirical results indicate that novice designers' overall performance increased 30.9~46.0% when using EIPW, and increased 33.5~34.9 % when using HBT, compared with the cases with no tools.

Keywords: HBT, EIPW, entity instance repository, relationship instance repository, conceptual data modeling, reuse pattern, ER model

1 Introduction

Conceptual data modeling is challenging because it requires good understanding of an application domain and an ability to translate requirements into a data model. Novice modelers frequently lack experience and have incomplete knowledge about the application domain being designed. Even expert designers could fail to obtain a quality conceptual model when they lack domain knowledge. Hence, data modelers at any level should benefit from reusing existing modeling knowledge during the modeling process. In addition, concepts that are not explicitly expressed in a requirement but are necessary for the domain are often very difficult to capture. Expertise in domain knowledge to identify entities and relationships hidden in the requirement is therefore also needed [1].

At present, a fully automated conceptual modeling approach seems impossible due to the inherent ambiguities in natural language (NL), context-dependent nature of modeling, and incompleteness of domain knowledge. When complete automation is not possible, it is still desirable to develop a semi-automatic process than an entirely manual modeling process. Therefore, many researchers have proposed knowledge-based systems (KBSs) and tools to support the modelers in developing conceptual models. One of the limitations of the proposed tools is that such tools do not solve the problems that novice modelers are inexperienced and have incomplete knowledge. In addition, they do not address the semantic mismatch issue [2], which represents the inability of novice modelers for translating the requirements literally into conceptual modeling structures.

Most conceptual designs are usually created from scratch, although a similar design might have previously been created. Reuse of 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 a major obstacle in facilitating reuse of knowledge [3]. One solution to reduce the 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 development of automated data modeling tools that help novice modelers develop quality conceptual data models. 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 via reverse engineering. The 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 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 is a popular technique for conceptual modeling because it could lead modelers with known heuristics. However, this approach does not provide an optimal solution to many complex requirements because most of the proposed heuristics/rules were built based on syntax of some specific NLs. These rules cannot overcome the inherent ambiguities of NLs. In this research, we test six data modeling rules that have been used in an introductory data modeling class for many years. We evaluate the usefulness of these rules by developing a tool named heuristic-based technique (HBT) that applies these rules to the identification of entities.

Two tools were developed in this study: HBT and EIPW (Entity Instance Pattern WordNet). The tasks of our tools are divided into two subtasks: the entity identification and the relationship identification. The entity identification processes of our tools are different, but the relationship identification processes in them are the same. The entity identification process of HBT incorporates the six domain independent modeling rules and entity categories adopted from Taxonomic Class

Modeling (TCM) [1] for identifying entities that were not explicitly stated in the requirement. The entity identification process of EIPW incorporates EIR, hypernym chains in WordNet, and entity categories adopted from TCM. The relationship identification processes of both tools incorporate RIR. WordNet is also used to ensure that the synonyms of EIR's entities and RIR's entities are not missed out. The architectures of each tool are discussed in later sections.

The goals of this study are as follows: (1) to find effective approaches that can improve the novice modelers' 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 tools are effective in creating quality data models.

The remainder of this paper is organized as follows. Section 2 presents five techniques for developing conceptual models. Section 3 provides a methodology to create EIR and RIR. Section 4 presents our six selected modeling rules. Section 5 presents the architecture of HBT. Section 6 presents the architecture of EIPW. Section 7 discusses the results from empirical experiments. Finally, Section 8 concludes the paper and gives directions for future work.

2 Related Techniques for Conceptual Modeling

There are at least five categories of techniques used for automatically generating conceptual models from NL requirement specifications. They are rule-based, pattern-based, case-based, ontology-based, and multi-techniques-based.

2.1 Rule-based

Chen [4] proposed eleven rules for translating English sentence structures into ER model's structure. Since then, many studies have tried to refine and extend on this approach. The trend in this technique orients towards the collaboration with huge linguistic dictionaries and common sense ontologies. The domain independence is the strength of this technique. However, the strength of this technique is also its weakness because tools or systems proposed have no domain knowledge incorporated in them. Most of tools proposed for developing conceptual models follow this technique [5].

2.2 Pattern-based

Recently, analysis patterns [6] have been popular and used in conceptual designs. The advantage of reusable patterns aims not only to reuse schema components, but also to reuse relationships between objects, which is a difficult task for novice modelers. However, building a repository of patterns involves explication of human developers' knowledge, which is a major obstacle in facilitating reuse of knowledge [3]. Another limitation of using this technique is that most of the available patterns in this field are analysis patterns that require manually matching.

2.3 Case-based

This technique involves storing the whole conceptual models of a large number of applications and providing a keyword mechanism that enables users to search for a conceptual model that is a candidate solution for a requirement [7]. It takes the advantage of reusing previous designs. The limitation in this technique is that if any

adjustment is required in the conceptual model, it has to resort to the generic data modeling approach. Another disadvantage is that developing the conceptual model libraries and indexing mechanism are very expensive.

2.4 Ontology-based

Ontologies have been considered as important components in many applications. Some generic and large scale ontologies such as WordNet, SUMO, and Cyc are available, but most applications require a specific domain ontology to describe concepts and relations in the domain [8]. Some researchers proposed the approaches with the development of conceptual models by refinement of large scale ontologies [9] [10]. However, currently there are no supporting tools or effective APIs to enhance the process. Most studies of ontology development and applications assume manual processes.

2.5 Multi-Techniques-Based

From our survey, most tools or systems for conceptual design require users' involvement during the process. And no single technique work best all the times because each technique has some limitations. Ideally, various techniques should be integrated together for a design process. For example, Song et al. [1] proposed a TCM (taxonomic class modeling) methodology used for object-oriented analysis in business applications. This method integrated several class modeling techniques under one framework. Their framework integrates the noun analysis method, class categories, English structures, check lists, and rules for modeling. In our study, we adopted the noun analysis method, class categories and modeling rules from the TCM work.

3 A Methodology for Creating EIR and RIR

This section presents our automatic methodology for creating the Entity Instance Repository (EIR) and the Relationship Instance Repository (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. An example of an EIP and an RIP is shown in Figure 1. We propose a method based on database reverse engineering concepts that are inspired by Chiang et al. [11] to automatically extract EIPs and RIPs from relational schemas. In this paper, we use the UML class diagram notation for representing ER models. This methodology employs three assumptions for the characteristics of input schemas for database reverse engineering process:

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

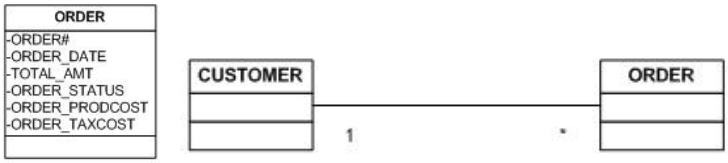

Figure 1. An example of an EIP and an RIP, respectively

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. In our study, we use the library of DDL schemas created by Silverston [12] containing 464 relations and 1859 attributes as our first input. Later the lists of EIR and RIR were extended by case studies.

(2) Extracting EIP's elements

We extracted 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 2.

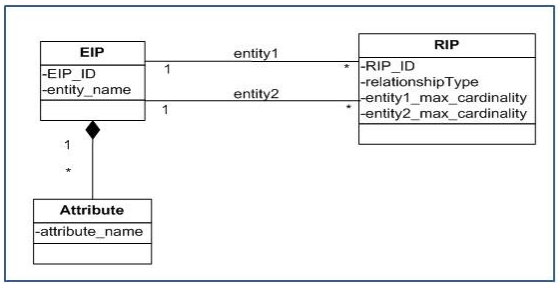

Figure 2. The metadata model of an EIP and an RIP

(3) Extracting RIP's elements

We extracted the RIP's elements by identifying relationships between extracted entities 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 [13]. 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 binary 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 an RIR. The reverse engineering rules used in this step are created by inverting the schema transformation rules based on the EER (Extended Entity-Relationship) approach [14]. The definitions of these transformation rules are described in [15].

4 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 [2] [16]. Also, 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 authors of this paper. These six rules are considered a minimal set of rules to teach novice designers in identifying entities. 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 is an entity.
R2: The MA (Multiple Attribute) Rule
IF a concept has multiple attributes, THEN it is an entity.
R3: The MVA (Multi-Valued Attribute) Rule
IF a concept has multi-values, THEN it is 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 is 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 is an entity.

5 Overview of HBT Architecture

The modules and data flow of HBT are shown in Figure 3. A prototype of HBT was developed by using 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 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 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 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 [1] . The discard noun set and the discard verb set are domain independent.

In the entity identification 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 the hidden entities that are not explicitly stated in the requirements but are necessary for the conceptual modeling by applying entity categories. Entity categories are domain knowledge and used as a tip for identifying candidate entities. Our entity categories in business domain are adopted from the class categories developed by Song et al. [1].

After, getting the entity list from Entity Identification Module, relationships between entities are generated by considering the application domain semantics inherent in the RIR. The relationship modeling rules [1] are used to ensure that all of the relationships are identified. WordNet is also used to ensure that the synonyms of RIR's entities are not missed out while preparing a list of the relationships. The lists of EIR and RIR are extended by case studies. Figure 4 shows one of the user interfaces of HBT.

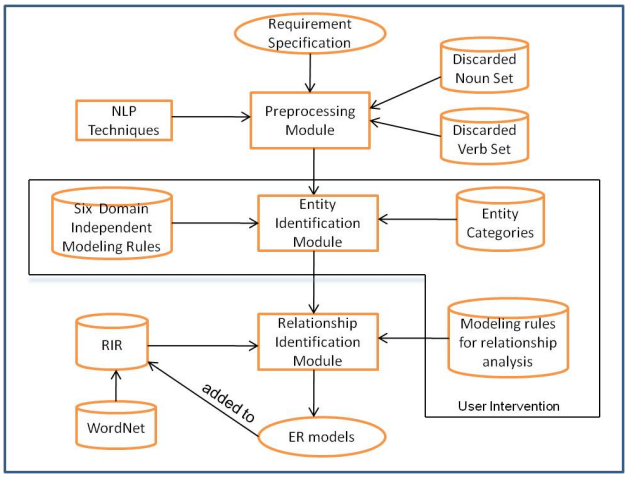

Figure 3. The HBT Architecture

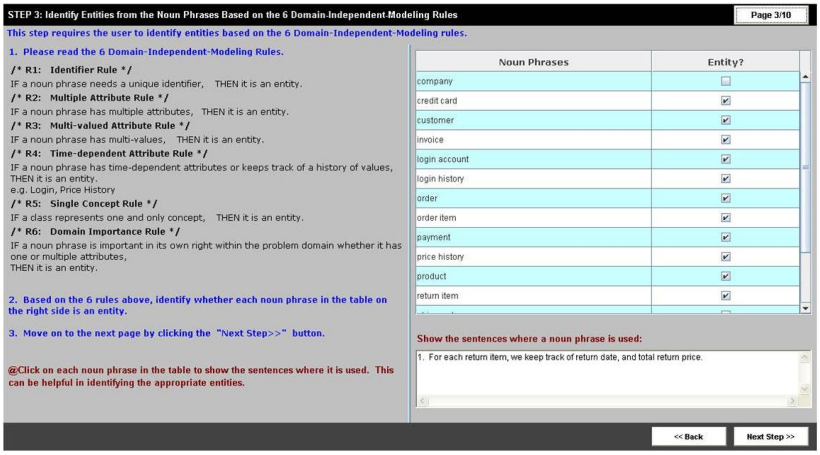

Figure 4. An example of user interface in HBT

6 Overview of EIPW Architecture

The system modules and data flow of EIPW are shown in Figure 5. A prototype of EIPW was developed by using JAVA Applet. Most of the modules' functions in EIPW are very similar to those in HBT. 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 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 [5].

(3) The third activity is to identify the hidden entities by applying entity categories.

The user interfaces of EIPW are also similar to those in the HBT as shown in Figure 6. The more details of the EIPW's workflow is presented in [15].

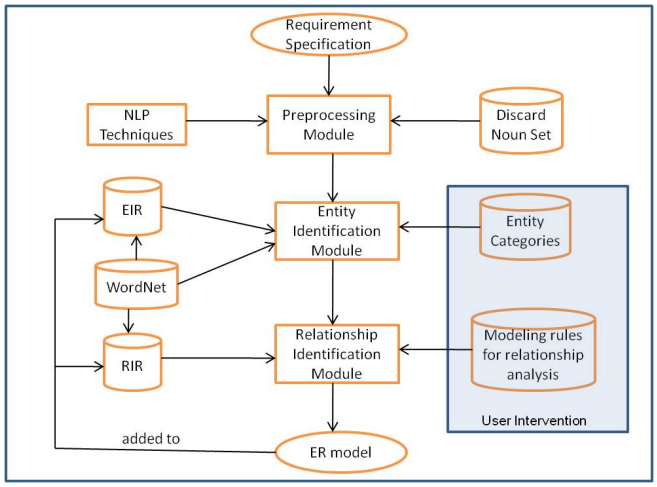

Figure 5. The EIPW Architecture.

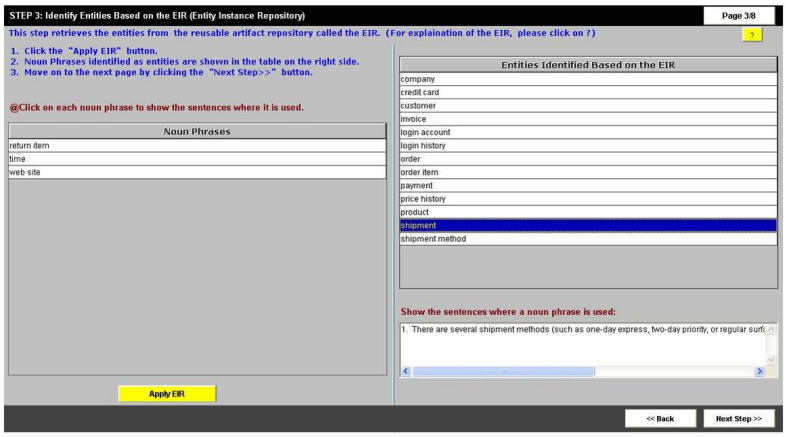

Figure 6. An example of user interface in EIPW.

7 Empirical Evaluation

In this section, we evaluated the quality of the output generated by our proposed KBSs by using the ANOVA technique. Since the quality of the conceptual 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 by novice designers without using any tools.
H2: Novice Designers using HBT will create conceptual models with better quality compared to the models generated by novice designers without using any tools.

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

7.1 Experimental Design

The experimental framework is shown in Figure 7. 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 [2]. As the task size increases, so do the numbers of decisions required in the modeling process. Therefore, our experimental 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 ERD. The accuracy of an ER model is evaluated by a scoring schema, which we adopted from Du [5]. It focuses on the correct identification of appropriate entities and relationships based on the given problem statements. The conceptual models created by the subjects are judged by third parties (not the authors of this paper). To take into account differences in task size, the quality scores obtained for each design are normalized in percentage.

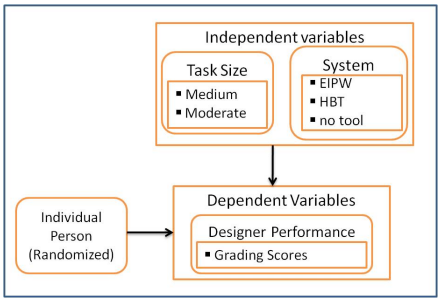

Figure 7. The framework of experiments

7.1.1 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 1. Each subject worked on four problem statements [17]: 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 1. The Experimental 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

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 2.

Table 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	$F(1,20) = 1.085$, $p < 0.310$

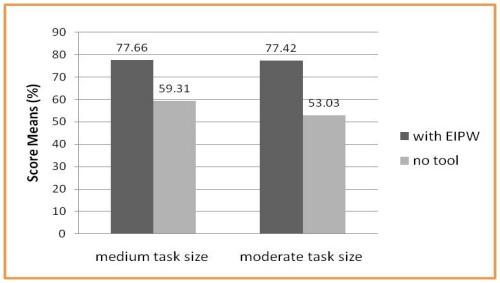

Figure 8. The plot of the mean quality scores (%)

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 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 3.

Table 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	$F(1,19) = 0.132$, $p < 0.720$

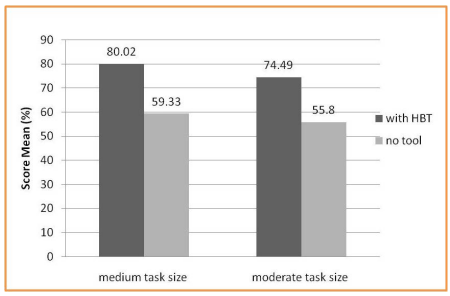

Figure 9. The plot of the mean quality scores (%)

From the calculated means shown in Figure 9, the conceptual models created by the HBT are better than those created by *no tool cases* for both task sizes. In Table 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.

Table 4. Tests of between-subjects effects with dependent variable QUALITY SCORE

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

In Table 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 medium to moderate task sizes.

8 Conclusions and Future Research

In this paper, we have proposed methods for improving the process of automatically developing conceptual data models. We have implemented two knowledge-based data modeling tools: HBT and EIPW. They use different techniques for entity identification, but use the same techniques for relationship identification. HBT uses noun phrases, verb phrases, identification of entities from hidden requirements using entity categories, and the six domain independent modeling rules. EIPW uses noun phrases, an entity instance repository (EIR), entity categories, and WordNet. Relationship identification uses a relationship instance repository (RIR), and WordNet. This study is an initial step to show how domain knowledge stored in the form of instance patterns can be used together with other modeling techniques.

The empirical results indicate that novice modelers' 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 modelers 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 rules in HBT are effective in identifying entities. They also minimize the cognitive load on the novices. 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 tools can ease the identification of relationships.

The study has, so far, been carried out on one domain only, but it provides a theoretical background for research on other domains as well. For future work, we want to test the usability of the tools for different domains and subjects. We plan to make our tools interface module to import the output schema into an ER diagram or a class diagram in commercial graphical CASE tools.

References

1. Song, I.-Y., Yano. K., Trujillo, J., Lujan-Mora, S.: A Taxonomic Class Modeling Methodology for Object-Oriented Analysis. In: Krostige T.H.J., Siau, K. (eds.) Information Modeling Methods and Methodologies. Advanced Topics in Databases Series, pp. 216-240. Idea Group Publishing (2004)
2. Batra, D.: Cognitive complexity in data modeling: causes and recommendations. Requir. Eng. 12(4), 231-244 (2007)
3. Han, T., Purao, S., Storey, V.: Generating large-scale repositories of reusable artifacts for conceptual design of information systems. Decision Support Systems 45, 665-680 (2008)
4. Chen, P.: English Sentence Structure and Entity-Relationship Diagram. Information Sciences 1(1), 127-149 (1983)
5. Du, S.: On the Use of Natural Language Processing for Automated Conceptual Data Modeling. Ph.D Dissertation, University of Pittsburgh (2008)

6. Purao, S., Storey, V., Han, T.: Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning. Information Systems Research 14(3), 269-290 (2003)
7. Choobineh, J., Lo, A.: CABSYDD: Case-Based System for Database Design. Journal of Management Information Systems 21(3), 242-253 (2004)
8. Sugumaran, V., Storey, V.: The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Trans. Database System 31(3), 1064-1094 (2006)
9. Conesa, J., Olivé, A.: A method for pruning ontologies in the development of conceptual schemas of information systems. Journal of Data Semantics 5, 64-90 (2006)
10. Conesa, J., Storey, V., Sugumaran, V.: Usability of Upper level ontologies: The case of ResearchSyc. Data & Knowledge Engineering, 69(4) (2010)
11. Chiang, R., Barron, T., Storey, V.: Reverse engineering of relational databases: Extraction of an EER model from a relational database. Data & Knowledge Engineering 12, 107-142 (1994)
12. Silverston, L.: The Data Model Resource Book Revised Edition Volume 2: John Willey & Sons Inc. (2001)
13. Song, I.-Y., Evans, M., Park, E.: A Comparative Analysis of Entity-Relationship Diagrams. Journal of Computer and Software Engineering 3(4), 427-459 (1995)
14. Elmasri, R., Nevathe, S.: Fundamentals of Database Systems. Redwood City, CA: The Benjamin/Cummings Publishing Co., Inc, Redwood City, CA (2004)
15. Thonggoom, O., Song, I.-Y., An, Y.: EIPW: A Knowledge-based Database Modeling Tool. In: CAiSE Workshops. (2011)
16. Parson, J., Saunders, C.: Cognitive heuristics in software engineering: applying and extending anchoring and adjustment to artifact reuse. IEEE Trans. Software Engineering, 30(12), 873-888 (2004)
17. Coronel, C., Morris, S., Rob, P.: Database Systems: Design, Implementation and Management: Course Technology. (2009)