EIPW: A Knowledge-Based Database Modeling Tool

Ornsiri Thonggoom, Il-Yeol Song, and Yuan An

The iSchool at Drexel University, Philadelphia, USA
{ot62,songiy,yuan.an}@drexel.edu

Abstract. 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 repository of reusable artifacts involves explication of human developer's knowledge, which is a major obstacle in facilitating reuse of knowledge. In this research, we explore knowledge-based and pattern-based approaches that help database designers develop quality conceptual data models based on reusable patterns. Our methodology includes database reverse engineering concepts. We propose new types of reusable instance patterns containing knowledge about an application domain, called entity instance repository (EIR) and relationship instance repository (RIR), which can be automatically generated from prior designs. The knowledge-based system (KBS) with EIR, RIR, and WordNet called EIPW (Entity Instance Pattern WordNet) is developed. The empirical study was conducted to test the efficiency of the KBS and indicated the significant improvement in novice designer performance.

Keywords: automate database design, reusable artifacts, patterns, conceptual modeling, entity-relationship modeling, ER diagram.

1 Introduction

The Entity-Relationship (ER) model [1] is one of the most well-known conceptual modeling formalisms. It is easy to understand, powerful to model real-world problems, and readily translated into a database schema. The ER model consists of a collection of entities, relationships between entities and attributes describing entities and relationships. Many studies [2, 3] have shown that conceptual modeling designs, especially developed by novice designers, may lead to inaccurate models.

Available commercial graphical CASE tools can be useful in documenting the output of analysis and design. However, they do not provide any supports 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 developing conceptual models. One of the limitations of proposed tools or KBSs 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 [4] and semantic mismatch [5].

Most conceptual designs are usually created from scratch, although similar design might have previously been created. And in many organizations there are a large

C. Salinesi and O. Pastor (Eds.): CAiSE 2011 Workshops, LNBIP 83, pp. 119–133, 2011.

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 systems development process. Currently, building a repository of reusable artifacts involves explication of human developers' knowledge, which is a major obstacle in facilitating reuse of knowledge [6]. It requires efforts from experts to identify elements with potential reuse, and then convert these into reuse elements. One solution to reduce the efforts and time of human experts comes from extracting the artifacts from the prior designs. If this could be conducted for various application domains, then it would assist in creating the practically reusable artifacts.

In this research, we explore knowledge-based and pattern-based approaches that help database designers develop quality conceptual data models. Our methodology 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. The patterns in EIR and RIR are also extended with the case studies. Our proposed artifacts are useful for conceptual designs in the following aspects: (1) they contains knowledge about an application domain; (2) automatic generation of EIR and RIR overcame a major problem of inefficient manual approaches that depend mainly on experienced and scarce modeling designers and domain experts; and (3) they are domain-specific and therefore easier to understand and reuse.

This paper aims at (1) proposing an automated methodology for creating the EIR and RIR as new types of reusable artifacts for conceptual modeling design which contain knowledge about an application domain; (2) developing an effective knowledge-based system with EIR and RIR called EIPW (Entity Instance Pattern WordNet), by integrating pattern-based technique and various modeling techniques; and (3) evaluating the usefulness of the KBS by human subject experiments.

The remainder of this paper is organized as follows. Section 2 summarizes the work on reusable artifacts related to our work, including the tools that incorporate with the reuse patterns. Section 3 presents the architecture of EIPW. Section 4 provides a methodology to create EIR and RIR. Section 5 describes the implementations of EIPW. Section 6 presents results from empirical testing of the EIPW. Finally, Section 7 concludes the paper and gives directions for future work.

2 Prior Research

This section presents the research on reusable artifacts related to our work, including the tools that incorporate with the reuse patterns.

2.1 Reusable Artifacts

To date, patterns have been well established as a technique for reusing solutions of recurrent problems in the software development process. Integrating patterns into

conceptual design is challenging. The recognition of patterns in conceptual data modeling field is based on works by Coad et al. [7], Hay [8], and Fowler [9]. Several authors have proposed various kinds of patterns [6-13]. However, their utility to conceptual modeling varies greatly. For the latest one, Blaha [3] 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; and canonical patterns are corresponding to meta models of modeling formalisms. Finally, he presents how to map his patterns to relational schema for database design.

There are the 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 [14] and [15]. However, packaged data models cannot replace the sound database analysis and design. Skilled designers are still needed to determine requirements and select, modify, and integrate any packaged models that are used. And no packaged data models provide any automated support for extracting the most suitable pattern for a particular situation. In this research, we created a library of patterns based on the package data models [14] and automated finding the most appropriate one for a certain situation, in which the reuse of the huge amount of knowledge saved in the patterns. As a first step, a pattern library included patterns from one domain only.

2.2 Tools and Systems Using Pattern Techniques

Analysis pattern repository has been the most popular and used in conceptual modeling tools or systems. To make use of patterns from a repository, the designer must be able to match a task description with a candidate pattern. Purao et al. [16] develop a KBS called APSARA, which automates analysis patterns to create object-oriented conceptual design. In his method, he first uses natural language processing (NLP) to parse the problem statement into significant keywords, and eventually objects. Based on object identification, analysis patterns are retrieved from the pattern base, then instantiated and synthesized into a conceptual model. In his pattern base, analysis patterns by Coad are used. Later, he augments his approach with a supervised learning concept [17]. The limitation of this KBS is that analysis patterns are so abstract that mismatches to patterns are fairly common. Wohed [18] proposes a Modeling Wizard Tool, which is a dialogue tool for selecting the appropriate patterns. Booking domain patterns are stored in the system. The appropriate pattern is selected step by step according to the answers given to the questions. However, this tool requires much more on user's interventions and it is hard to use it for large scale batch processing.

3 Overview of EIPW Architecture

The system modules and data flow of EIPW are shown in Figure 1. A prototype of EIPW has been developed by using Java Applet. Firstly, the system passes 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. Then, The EIR and the hypernym chains in

WordNet [19], and entity categories which are knowledge repositories in an application domain are used for the entity identification. Our entity categories in business domain are adopted from the class categories developed by Song et al. [20]. WordNet is also used to ensure that the synonyms of EIR's entities are not missed out while preparing the list of candidate entities. After, getting the entity list from Entity Identification Module, relationships between entity list 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.

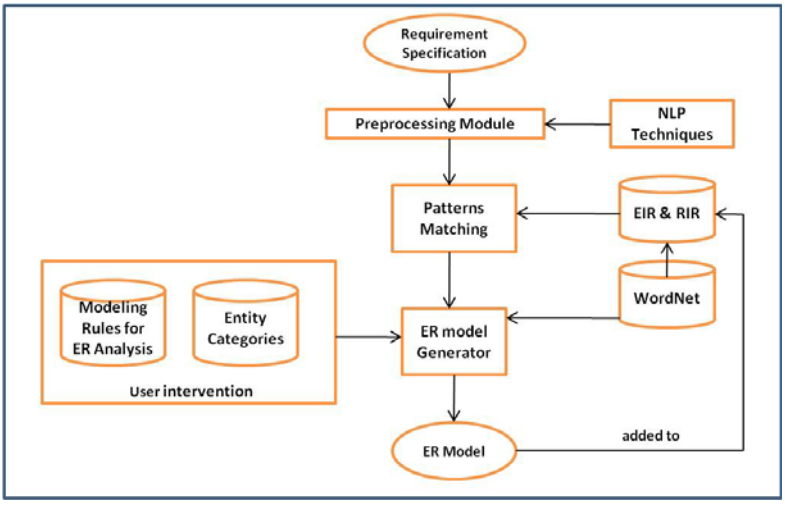

Fig. 1. The EIPW Architecture

4 Methodology for Creating EIR and RIR

This section presents the automatic methodology for creating EIR and RIR, which are the repositories of EIPs and RIPs, respectively. These repositories 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. We propose a method based on a database reverse engineering concept and were inspired by Chaing et al. [21] to automatically extract EIPs and RIPs from relational schemas. In this paper, we use UML class diagram notation representing the ER models. This methodology employed three assumptions involving the characteristics of the 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): PKs or FKs are specified for every relation of the input DDL schemas.

An example of an EIP and an RIP is shown in Figure 2.

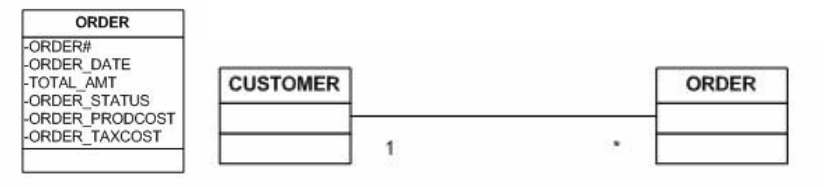

Fig. 2. An example of an EIP and an RIP, respectively

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. These executable schemas have to provide at least relation names, attribute names, and PKs. In our paper, we use the library of DDL schemas created by Silverston [14] containing 464 entities 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 extracted the EIP's elements from the 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.

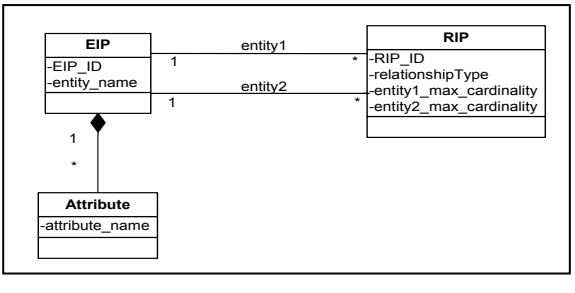

Fig. 3. The Meta-Model of EIP and RIP

(3) Extracting RIP's elements
We extracted the RIP's elements by identifying relationships between extracted entities from Step (2) above. In this research, we only specified the default cardinality ratios for binary models. Most of the ER methods used in textbooks or CASE tools

can be classified as either binary models or n-ary models [22]. Because of the limited semantic expressiveness of DDL schemas, the minimum cardinality cannot be automatically identified. For totally automatic 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

Then, these binary relationships are stored in RIR. The reverse engineering schema transformation rules used in this step were created by inverting the traditional concepts of database design based on EER approach [23]. 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 coming from T_1, then there is a 1: N relationship between T_1 and T_2.

3.2 1: N for relationships identified by partial keys
IF: the PK of a relation T_1appears as a composite PK of another relation T_2 and the PK of relation T_1 is the FK of table T_2 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, T1 is a strong entity, T_2 is a weak entity, and there is a 1: N relationship between T_1 and T_2.

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.

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})

3.5 Recursive Relationship
IF: T1 has a FK that references its own table (T),
THEN: T has recursive relationship.
Consider this relation:

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

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

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

5 The EIPW

This section shows the workflow of EIPW and its use for generating an ER model. In the previous section, EIR and RIR have been created as the reusable pattern repositories. This section discusses how these patterns will be implemented. EIPW can be divided into two subtasks: entity identification and relationship identification.

1. Entity Identification
The actual step-by-step activities of our methodology outlined in Figure 4 are in the form of an activity diagram in the UML.

In Figure 4, the three swimlanes perform the following activities:

- The middle swimlane: The aim of these swimlane activities is to identify the entities based on EIR.
- The rightmost swimlane: The aim of these swimlane activities is to identify the entities that are not detected by EIR by applying the 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. Our entity categories in business applications adopt the class categories created by Song et al. [20]. Entities categories are used as a tip for identifying entities.

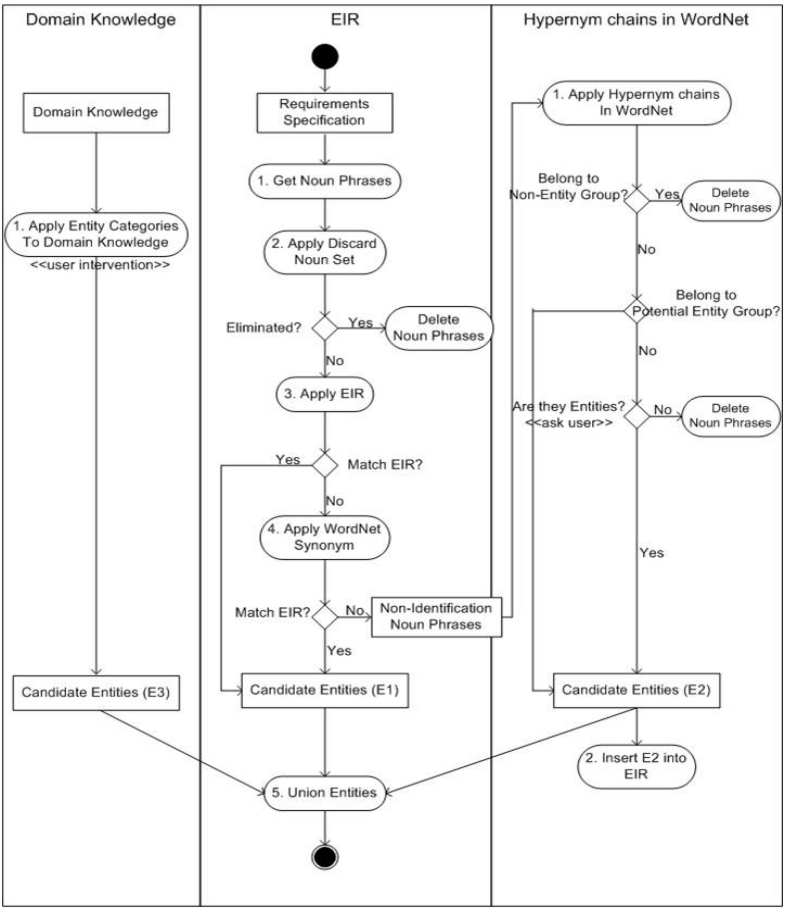

Fig. 4. Entity Identification Process in EIPW

The details of the activities in Figure 4 are presented below.

Activities of Middle Swimlane of Figure 4

- Begin with a requirement and remove the partial explanation statements

Explanation statements in a requirement specification aim to help human to understand the requirement better but they are harmful for automated requirement analysis [8]. Heuristics based on parenthesis and some words (e.g. such as) were used to remove the explanation statement.

- Step 1: Get noun phrases.

The Part of speech (POS) tags provide the word's syntactic categories of words whereby candidate entities can be identified from either noun phrases [5] or verb phrases [22]. In this research, we used a well-known open source called LingPipe (http://alias-i.com/lingpipe) for POS tagging to get all the noun phrases from requirement specification. Figure 5 shows the user interface of Step 1 that lists all the noun phrases appearing in the requirement specification.

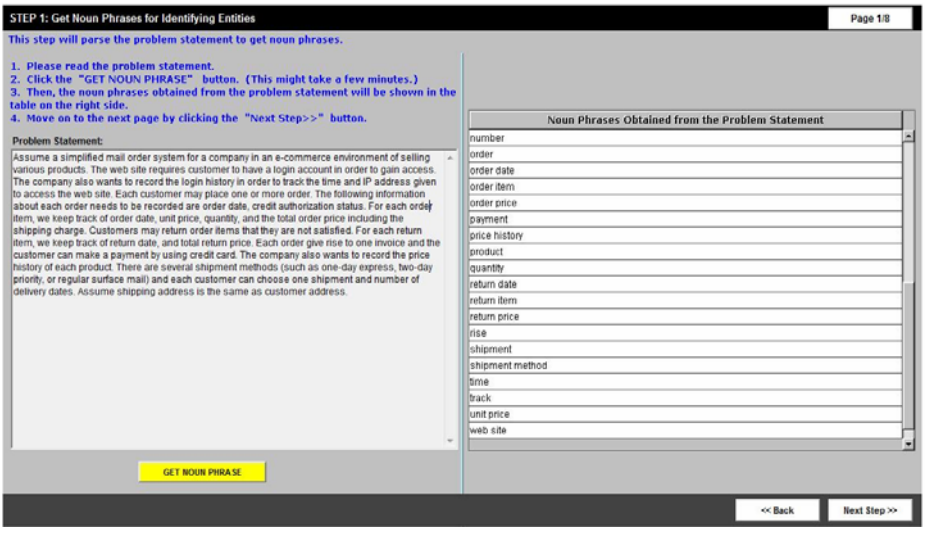

Fig. 5. The user interface of Step 1 listing all noun phrases appearing in the requirement

- Step 2: Test the discard noun set.

To facilitate the post-parsing analysis, noun phrases belonging to any of the discard noun set will be deleted. The discard noun set was created based on the history of words discarded by designers and the class elimination Rule [22]. The discard noun set is domain independent. The examples of nouns in discard noun set are number, ID, information, database, track, record, and system.

- Step 3: Identify entities based on EIR

Each noun phrase gotten from Step 2 was compared to entity names in the EIR. If a noun phrase matches an entity name in EIR, then the noun phrase 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 the noun phrase match an entity name in EIR, then the noun phrase becomes a candidate entity (E1).

Activities of RightMost Swimlane of Figure 4

- Step 1: Apply hypernym chains in WordNet

They are used to perform entity categorizations. The entity categories can help identifying 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 any groups, 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 4

In this swimlane, the system asks the users to identify the hidden entities by applying domain knowledge to entity categories. In this process, we adopted the class categories created by Song et al. [22] as the entity categories.

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

For each entity category, check whether all the entities representing the entity categories are already captured. Otherwise create a new 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: $\{E\} = \{E1\} \cup \{E2\} \cup \{E3\}$

2. Relationship Identification

After the entity list has been identified in the entity identification process, relationships between entity list 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 relationship between entities. The flowcharts for the relationship identification are shown in Figure 6. In Figure 6, 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 user to identify the relationships, which are not detected by the RIR, by applying Need-to-Remember Rule [20].

The details of the activities in Figure 6 are discussed below.

Activities of left swimlane of Figure 6

- Begin with the candidate entity list gotten from entity identification process.
- Step 1: Delete duplicate entities.
  This will be conducted through WordNet synonyms.
- Step 2: Assign all the 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.

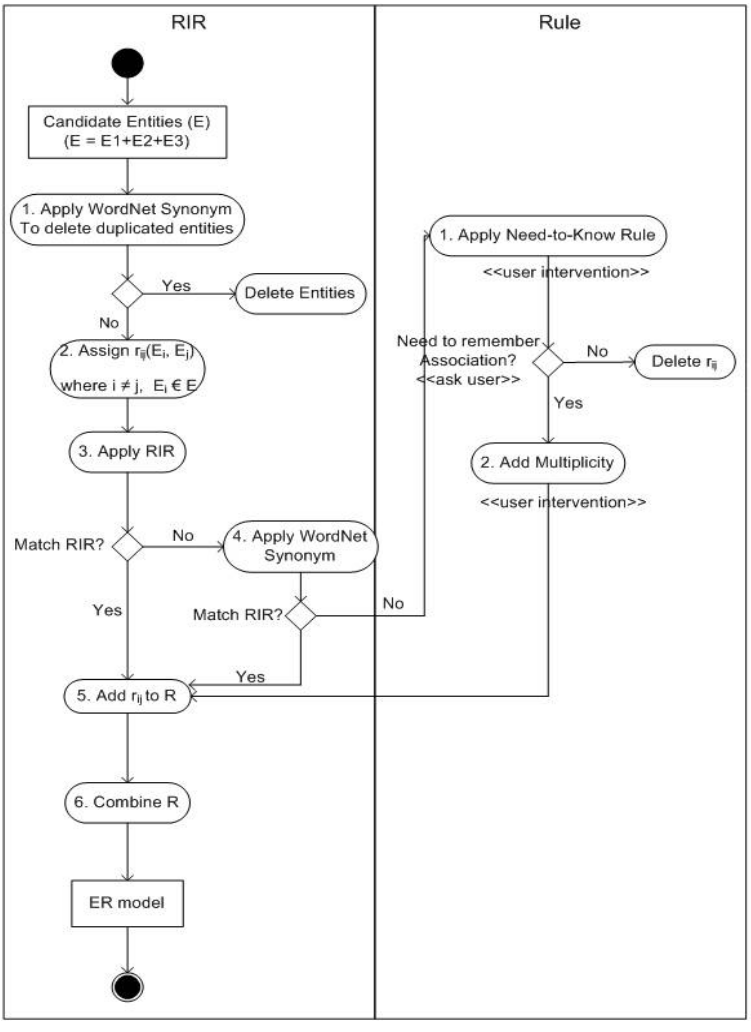

Fig. 6. Relationship Identification Process in EIPW

Activities of right swimlane of Figure 6

- Begin with the possible relationships that are not detected by the RIR from left swimlane.
- Step 1: Apply Need-to-Know rule (user intervention)
  If a relationship represents an association that does not have to be remembered between two entities, then eliminate this relationship.
- Step 2: 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 7 shows the output of EIPW.

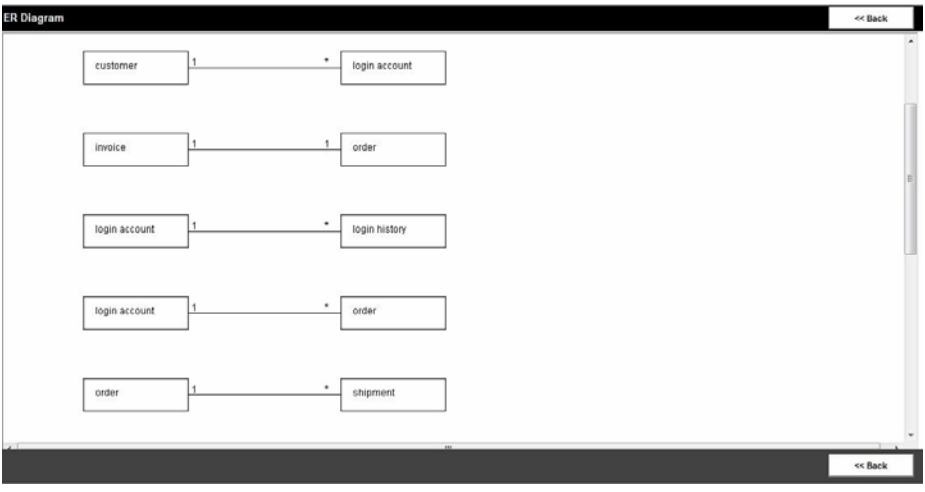

Fig. 7. The Output of EIPW

6 Empirical Evaluation

In this section, we evaluate the quality of output generated by EIPW by using an ANOVA technique. The resulting ER model created with our EIPW would be of higher quality than the one created with no tool. Since the quality of the ER models is of interest, the following hypothesis is tested:

H1: Novice Designers using EIPW will create ER models with better quality compared to the models generated by designers without the use of the system.

6.1 Experiment Design

We used an ANOVA technique called *2x2 within-subjects (repeated-measures) design*. The two independent variables are the system (with the aid of EIPW and no tool) and the task size (medium, moderate). Based on the combinatorial complexity [5], as the task size increases, so do the numbers of decisions required in the modeling process. Therefore, our experiment design incorporated 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. In this research we adopted the grading scheme proposed by Du [19]. It focuses on the correct identification of appropriate entities and relationships based on the given problem statements. The ER models created by the subjects were judged by a third party (not the authors of this paper).

6.1.1 Subjects and Tasks

There were 20 subjects. All of the subjects were students in the iSchool at Drexel University and did not work in database design field before. Therefore, we concluded that

all of our subjects were novice designers. Eight are undergraduates and twelve are graduate students. Twenty subjects were divided into two groups as shown in Table 1. Each subject worked on four problem statements: one medium size and one moderate size problem statements with the aid of EIPW and one medium size and one large moderate problem statements with no tool. The problem statements were in an e-commerce domain. This domain was not too familiar to the students (unlike e.g. university management and library domains), and thus there was no concern that students would answer the questions due to their background experiences. The subjects can take time as long as they wanted to create ER models based on the given problems.

Table 1. The Experiment Design

Group	Num of subject	Problem1	Problem2	Problem3	Problem4
Group 1	10	No tool	No tool	Using EIPW	Using EIPW
Group 2	10	Using EIPW	Using EIPW	No tool	No tool

6.2 Results

A 2x2 within-subjects analysis of variance was performed on ERD quality scores as a function of EIPW (with, no tool) and task sizes (medium, moderate) as shown in Table 2. Considering the differences between the task sizes, the quality scores obtained for each design were calculated in percentage.

Table 2. Test of between subjects with dependent variable QUALITY SCORE

	QUALITY SCORE
System (EIPW, no tool)	$F(1,19) = 96.01, p < 0.000$
Task Size (medium, moderate)	$F(1,19) = 33.01, p < 0.094$
System x Task Size	$F(1,19) = 1.06, p < 0.317$

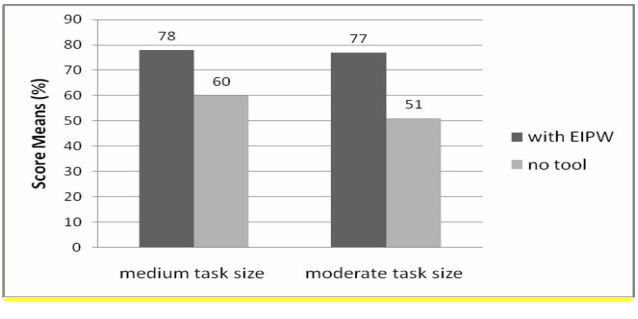

Fig. 8. The plot of the mean quality scores

From the calculated means shown in Figure 8, the ER models created by the EIPW are better than those created by no tool for each task size. From Table 3, the result shows that the main effect of system (with EIPW, no tool) was significant

($p < 0.00$). Therefore, this result supported our hypothesis (H1) that the EIPW helped subjects create better ER models than they do without it. There was no significant main effect for task size ($p < 0.094$). There was no significant system by task size ($p < 0.317$). It suggested that the main effect of the system (with EIPW, no tool) was not significantly different at the two levels of task size.

7 Conclusions and Future Research

In this paper, we have proposed a method for improving the process of conceptual modeling design. This research is an initial step to show how domain knowledge stored in the instance patterns can be reused for the conceptual modeling design. The EIPW, which integrated pattern-based technique and various modeling techniques, clearly helped the designers in creating better quality ER models. The empirical results indicated the significant improvement in novice designer performance when using the tool. The tool can help novice designers who are less experienced and who have incomplete knowledge in an application domain. The initial results suggested that our instance patterns can indeed be an asset in supporting conceptual modeling process because they minimize the cognitive load on the designers and ensure that the ER models are correct. In the educational context, our KBS can serve as the learning tool. It simplifies the work of experienced designers and provides a smooth head start to novice. However, 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 the future work, we want to test the usability of the KBS for different categories. For example, we want to repeat the experiment using the expert designers as the subjects. And we plan to make our EIPW interface module to import the output schema into an ER diagram or a class diagram in commercial graphical CASE tools.

References

1. Chen, P.: The Entity-Relationship Model: Toward A Unified View of Data. ACM Transactions on Database Systems 1(1), 9–36 (1976)
2. Moody, D.: Theoretical and Practical Issues in Evaluating the quality of conceptual models: Current State and Future Directions. Data & Knowledge Engineering 55, 243–276 (2005)
3. Simsion, G.: Data Modeling Theory and Practice. Technics Publications, LLC (2007)
4. Kim, N., Lee, S., Moon, S.: Formalized Entity Extraction Methodology for Changeable Business Requirements. Journal of Information Science and Engineering 24, 649–671 (2008)
5. Batra, D.: Cognitive complexity in data modeling: causes and recommendations. Requir. Eng. 12(4), 231–244 (2007)
6. 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)
7. Coad, P., North, D., Mayfield, M.: Object Models – Strategies, Pattern, & Applications. Yourdon Press, Englewood Cliffs (1995)
8. Hay, D.C.: Data model patterns: Conventions of Thought. Dorset House Publishing, New York (1996)

9. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison Wesley, Menlo Park (1997)
10. Batra, D.: Conceptual Data Modeling Patterns: Representation and Validation. J. Database Manag. 16(2), 84–106 (2005)
11. Blaha, M.: Patterns of Data Modeling. CRC Press, Boca Raton (2010)
12. Fayad, M., Schmidt, D., Johnson, R.: Object-oriented Application Frameworks: Problem and Perspectives. Willy, NY (1997)
13. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley, Reading (1998)
14. Silverston, L.: The Data Model Resource Book Revised Edition, vol. 2. John Willey & Sons Inc., New York (2001)
15. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Data Modeling. John Wiley & Sons, Inc., New York (2002)
16. Purao, S.: APSARA: A tool to automate system design via intelligent pattern retrieval and synthesis. Database Advance Information Systems 29(4), 45–57 (1998)
17. 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)
18. Wohed, P.: Tool Support for Reuse of Analysis Patterns - A Case Study. In: The 19th Int. Conf. on Conceptual Modeling (2000)
19. Du, S.: On the Use of Natural Language Processing for Automated Conceptual Data Modeling. Ph.D Dissertation, University of Pittsburgh (2008)
20. 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)
21. 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)
22. 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)
23. Elmasri, R., Nevathe, S.: Fundamentals of Database Systems. The Benjamin/Cummings Publishing Co., Inc., Redwood City,CA (2004)