Methodologies for Semi-automated Conceptual Data Modeling from Requirements

Il-Yeol Song[1(✉)], Yongjun Zhu[1], Hyithaek Ceong[2], and Ornsiri Thonggoom[3]

[1] College of Computing and Informatics, Drexel University, Philadelphia, 19104, USA
{song,zhu}@drexel.edu
[2] Department of Multimedia, Chonnam National University, Yeosu 550-749, Korea
htceong@chonnam.ac.kr
[3] Department of Textile Science and Technology, Thammasat University,
Bangkok 10200, Thailand
ornsirithonggoom@gmail.com

Abstract. Conceptual modeling is the foundation of system analysis and design methodologies. It is challenging because it requires a clear understanding of an application domain and the ability to translate the requirement specification into a conceptual data model. Semi-automated conceptual data modeling is a process of using an intelligent tool to aid the modeler for the purpose of building a quality conceptual data model. In this paper, we first present six categories of methodologies that can be used for developing conceptual data models. We then describe the characteristics of each category, compare these characteristics, and present related work of each category. We finally suggest a framework for semi-automatically generating conceptual data models from requirements and suggest challenging research topics.

1 Introduction

Conceptual modeling is the foundation of analysis and design methodologies for the development of information systems. For many years, researchers have proposed various conceptual data modeling formalisms such as Entity-Relationship (ER) model [13], variations of the ER model such as IDEF1X, Oracle CASE notation, and Information Engineering, Natural/Nijssen Language Information Analysis Method (NIAM), Extended Entity Relationship (EER) models, Object Role Modeling (ORM), object-oriented (OO) modeling, Unified Modeling Language (UML), EXPRESS, RM-ODP, and others. Thalheim [64] estimates that the total number of proposed variations of the ER model is over 80. Comparisons of these modeling formalisms are shown in studies by Song et al. [58], Kim and March [37], and Neill and Laplante [50].

Among the different conceptual modeling formalisms, ER models and UML models are the most widely used in practice [40]. The ER model originally proposed by Chen [13] has been widely used in systems analysis and conceptual modeling. The ER approach is easy to understand, models real-world concepts, and readily translates an ER diagram into a database schema [21]. Many extensions or variations of the ER model in the form of EER (Extended ER or Enhanced ER) model have been proposed and utilized in different

applications [28]. The UML class diagram is another widely used conceptual modeling approach, especially in software engineering.

Natural language (NL) is the most common tool for people to describe things and communicate. Studies [40, 50] show that nearly 90 % of all the requirements in industrial practices are written in NL. There are at least three limitations in translating a requirement in NL into a conceptual model [59]: (1) NL is ambiguous and an effective and accurate analysis is difficult; therefore, techniques and rules for modeling are required. (2) The same semantics can be represented in multiple ways; therefore, ways of handling these context-dependent semantic variations are necessary. (3) Concepts that are not explicitly expressed in a requirement specification but still important in the application are often difficult to discover and model. Expertise in domain knowledge to discover the hidden entities that were not explicitly stated is therefore needed.

Thus, this paper focuses on how to effectively translate a requirement specification into a conceptual model, wherein we review literature to find out the state-of-art in semi-automated conceptual data modeling.

The rest of this paper is organized as follows: Sect. 2 describes difficulties in creating conceptual data models; Sect. 3 discusses six categories of semi-automated conceptual data modeling methodologies; Sect. 4 proposes the next steps for semi-automated conceptual modeling as well as a framework for semi-automated conceptual modeling; and Sect. 5 concludes the paper.

2 The Difficulties in Creating Conceptual Data Models

The difficulties in creating conceptual data models have been discussed in a previous study [7]. In spite of its importance, it is shown that conceptual data modeling is not done well [56]. Through a literature review, we identified the factors that contribute to the difficulties of creating conceptual data models.

2.1 Combinatorial Complexity in Possible Relationships

A previous study [67] shows that novice designers experience more difficulties in modeling relationships than entities. Batra [7] reports that novice designers not only have difficulties in modeling unary and ternary relationships, but also have difficulties in modeling all kinds of meaningful binary relationships. As the number of entities increases, the number of possible relationships increases at a combinatorial rate. The primary challenge in modeling relationships is how to select a minimum set that captures the semantics effectively. For the meaningful identification of relationships, at least the following criteria should be met: (1) all semantics in the application should be modeled, (2) any relationship constructs should not be redundant, (3) and the degree of relationships should be minimal.

2.2 Scattered Modeling Rules

There is no single complete set of heuristics/rules that can be used in developing quality data models. In general, heuristics/rules are often useful, but sometimes they may lead

to cognitive errors called biases [53] if they are not applied appropriately. Furthermore, there is always a trade-off in applying heuristics: applying more heuristics requires less human intervention while more complexities arise. Context-dependent conflicts among heuristics could be one kind of complexity.

2.3   Semantic Mismatch

Translating a requirement specification literally into a database structure could cause literal translation errors [7]. For instance, a sentence stating that "an order records a sale of products to customers" may incur an erroneous relationship between customer and product entities. The example shows that not all real-world relationships map into meaningful database relationships in the data model; some real-world relationships are derivable at the database level.

2.4   Inexperience/Incomplete Knowledge of Designers

Novice designers have limited knowledge and skills, while expert designers often draw their knowledge from past experiences. Even expert designers might fail to create a quality conceptual model due to their lack of domain knowledge, unless he/she has a clear perception on the requirement specification [36]. Expertise in domain knowledge to identify the hidden entities is therefore needed. Important issues are how novice designers can be efficiently trained and how domain knowledge can be effectively transferred to the designers.

2.5   Multiple Solutions

In conceptual design, there is no single best answer. Moody and Shanks [48] also state that one of the common problems encountered in design is a large number of alternative designs can be created for a particular problem. Therefore, they propose a six-element framework to evaluate the quality of conceptual data models [47]. Their framework is composed of the following six factors: completeness, simplicity, flexibility, understandability, integration, and implementability. Later this framework is increased into eight factors by including correctness and integrity by empirically validating the framework [49].

3   Methodologies for Semi-automated Conceptual Data Modeling

The field of conceptual data modeling has spawned numerous techniques for the identification of entities and their relationships. Most of these techniques heavily rely on manual processes and experiences of designers. Currently, there are several commercial graphical CASE tools used for automatically converting a conceptual data model into a logical model and then generate a DDL definition in SQL for physical implementation. However, there is still no commercial tool that directly translates a NL requirement specification into a conceptual data model. At present, a fully automated conceptual

modeling approach seems extremely challenging due to the inherent ambiguities in NL, the context-dependent nature of modeling, and the incompleteness of domain knowledge in tools. It is desirable to develop a semi-automated intelligent tool that can alleviate those problems to assist modelers.

This section presents a broad scope of methodologies that can be used for semi-automatically developing conceptual data models. We classify them into six categories: linguistic-based, pattern-based, case-based, ontology-based, metamodeling-based, and multi-techniques-based.

3.1   The Linguistic-Based Approach

Natural language processing (NLP) techniques and linguistic theories are used for designing conceptual models. Chen [12] proposes eleven rules for translating English sentence structures into ER constructs. Since then, several studies [33, 51] have tried to refine and extend the approach. However, these rules are still neither fully complete nor accurate. Although entities can be identified by nouns in a requirements specification, not all the nouns are entities because nouns not only refer to entities but also refer to attributes, or other throw-away concepts which need not be modeled. An entity can also be identified from a verb phrase and implicit requirements [59]. Hartmann and Link [33] modified Chen's eleven rules for translation from English sentence structures into EER constructs in which they re-organize and extend those rules using twelve heuristics. However, even these heuristics are not complete. These rules cannot overcome the inherent ambiguities of NLs. In addition, these kinds of rules cannot be universally applied to multiple domains. Therefore, this approach can only serve as a basis for a manual or a semi-automated process of transforming an English specification into an ER diagram.

In order to overcome inherent ambiguities in NL requirements, some studies put constraints on the input by restricting either the vocabulary or sentence structures [3]. With these restrictions, simple linguistic processing such as tagging and chunking can achieve reasonably good results. However, the use of controlled languages has some limitations. They cannot be easily applied to existing requirement documents. Furthermore, they are not natural and place extra burdens on requirements writers. Several formal specification languages such as Z, Object-Z, VDM, B, and OCL have also been proposed for formal model-based specifications. They are very expressive but require formal knowledge of the language to write a correct specification. Moreover, these languages have been designed for some specific applications, and their usages for different purposes may become awkward and difficult. Other researchers propose dialogue tools that help elicit the NL requirement specification [9, 36]. The main disadvantage of these tools is that they require more laborious human interventions, and thus it is difficult to use them for large-scale projects. In [54], a requirement specification is rewritten based on a constrained grammar. This approach, however, could introduce semantic loss due to imposed constraints and requires the participation of experts who know the grammar.

Classification theory and entity categories have been applied to conceptual data modeling [59]. Entity categories are characterized by the properties shared by their

members and a widely-used technique in identifying entities or classes. In addition, entity categories can be used to discover missing or hidden entities or classes that are not explicitly stated in the requirement but can be discovered by applying domain knowledge to the entity categories [59].

A trend in this technique is the collaborative use of huge linguistic dictionaries [23, 45] and common sense ontologies [39]. Linguistic dictionaries provide not only semantic links among concepts such as synonym, antonym, hypernym (is-a), and meronym/ holonym (part-of), but also syntactical and morphological information. A variety of relationship types is discussed in [62]. WordNet [44] is a popular linguistic dictionary used for concept identification during conceptual modeling because it is readily available and extends to other languages such as European languages, Spanish, Chinese, and so forth. However, the main drawback of WordNet is that it does not cover many relationships. Therefore, WordNet++, the extension of WordNet, which contains special types of relationships that are not available in WordNet, is proposed in [19].

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

3.2 The Pattern-Based Approach

The important role of patterns in design is recognized in Alexander's book [4] on architecture and urban planning. It suggests that designers should produce and use patterns rather than solve 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 a reduction of time and cost in 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. The recognition of patterns in the context of conceptual data modeling is based on works by Coad et al. [15], Hay [34], and Fowler [26]. They created a library of proven modeling structures and provided some examples of adapting generic models to suit particular requirements.

Several authors have proposed various kinds of patterns [15, 26, 34]. Blaha [8] 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 metamodels of modeling formalisms. However, their utility in automated data modeling remains to be seen.

Empirical research shows that experts reuse patterns while novices do not [11].

The process in pattern reuse can be divided into three tasks: retrieval, adaptation, and integration [5]. Retrieval involves choosing patterns that may be relevant to a particular problem. After a pattern is chosen, it needs to 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. The advantage of reusable

patterns aims to reuse not only schema components but also relationships between objects. However, building a repository of patterns involves the explication of human developers' knowledge, which is a major obstacle in facilitating the reuse of knowledge. To develop a pattern repository, designers should have very clear knowledge about the specific domain and 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 a pattern repository. Currently, most of the proposed reusable pattern repositories used for conceptual data modeling are 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 that require manual matching.

One solution to reduce the effort and time of human experts comes from extracting the pattern artifacts from existing designs [31]. If this could be done for various application domains, then it would assist in creating practically reusable pattern artifacts.

3.3   The Case-Based Approach

Case-based reasoning is a technology that utilizes a repository of cases for decision-making. The basic idea is, given the description of a new problem, to retrieve a similar problem from the case repository and adapt the retrieval to obtain a solution.

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 [63], DES-DS [52], and CABSYDD [14]. A comparison between these KBSs can be found in [14].

This technique involves storing 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 problem statement. The technique takes advantage of reusing the previous design. The limitation of this technique is that if any adjustment is required in the conceptual model, it has to resort to the generic data modeling approach. Moreover, adjustments on the conceptual model are always required in order to cope with changes in requirement specification. The major disadvantage of this technique is that developing the conceptual model libraries and indexing mechanism are very expensive.

3.4   The Ontology-Based Approach

Ontologies have been proposed as an important way to represent real-world knowledge and, at some level, to support interoperability [57]. An ontology can represent in the form of a taxonomy, a thesaurus, a domain model, or a logical theory. Some papers point out some similarities and differences between ontologies and conceptual data models [20, 25]. According to Fonseca [25], two criteria that differentiate ontologies from conceptual data models are (1) the objective of modeling and (2) objects to model. Embley [21] suggests that ontologies are the key for solving the semantic problems of information systems.

In the conceptual modeling field, many researchers employ ontologies for evaluating, improving or developing conceptual modeling formalisms. Storey [61] proposes an ontology to classify the verb phrases of relationships based on research in linguistics

and semantic data models. Evermann and Wand [22] 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 [55] propose a multilayered ontology for classifying relationships by using data abstractions and by separating domain-dependent and domain-independent aspects of the relationship constructs.

The major advantage of using ontologies for conceptual modeling is the reusability of a knowledge repository. This can be developed into two levels: domain ontology and large scale or upper level ontology. A domain ontology [29] represents concepts, relationships between concepts, and inference rules for a particular domain. Several tools for creating and querying domain ontologies are available such as Protégé, OWL, SPARQL, etc. A detailed comparison of each tool is shown in [17]. Instead of developing individual ontologies, there has been interest in creating an upper level or large scale ontology. An upper level ontology [29] 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 a review and comparison of upper level ontologies see [42]. The potential usefulness of upper level ontologies lies in the fact that they are domain independent. 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 [16]. Without the support of an adequate tool, it is difficult to work with them. Obviously, domain ontologies are more usable than large scale ontologies [29].

An ontology can be used as the source of domain knowledge, and designers can use the ontology to get initial domain knowledge. Corcho and his colleagues [17] suggest that a strategy for developing domain ontologies would be to reuse large scale or upper level ontologies. The same upper level ontology can be used for developing many domain ontologies, which share the same skeleton. Extensions of the skeleton should be made at a low level by adding domain-specific subconcepts.

3.5   The Metamodeling Approach

A metamodel describes the syntax of the models with a collection of modeling concepts and constraints. Metamodeling is an abstraction-based modeling technique; it is a process of creating models using the domain knowledge starting from the metamodel. One of the most active research areas utilizing metamodeling is model-driven development [6]. Metamodeling allows the user to use permitted structures and assign semantics of the domain to the structures [60]. Fill and Karagiannis conceptualize the components of modeling methods using the ADOxx metamodeling platform [24]. A detailed discussion of research challenges in metamodeling is presented in [60].

3.6   Multi-Techniques-Based Approaches

From our survey, most tools or systems for conceptual design require users' involvement during the process. No single technique works best all the time because each technique has some limitations. Ideally, various techniques should be integrated together during a modeling process. For example, Song et al. [59] have proposed a TCM (Taxonomic Class Modeling) methodology used for OO analysis in business applications. This method integrates several class modeling techniques under one framework. The framework integrates the noun analysis method, class categories, English structures, check lists, and modeling rules. Thonggoom et al. [65, 66] present two knowledge-based modeling tools that integrate the pattern-based technique, WordNet, and other heuristics-based modeling techniques. These tools show how domain knowledge stored in instance patterns can be reused together with other modeling techniques and improve the effectiveness of data modeling.

4   A Proposal for Semi-automated Conceptual Modeling

In this section, we propose a 3-step approach for semi-automated conceptual data modeling from requirement specification. The method includes the process of (1) analysis and refinement of the requirement specification, (2) generation of a conceptual model from the requirement, and (3) verification and validation of the generated model. Figure 1 shows our framework for semi-automated conceptual data modeling.
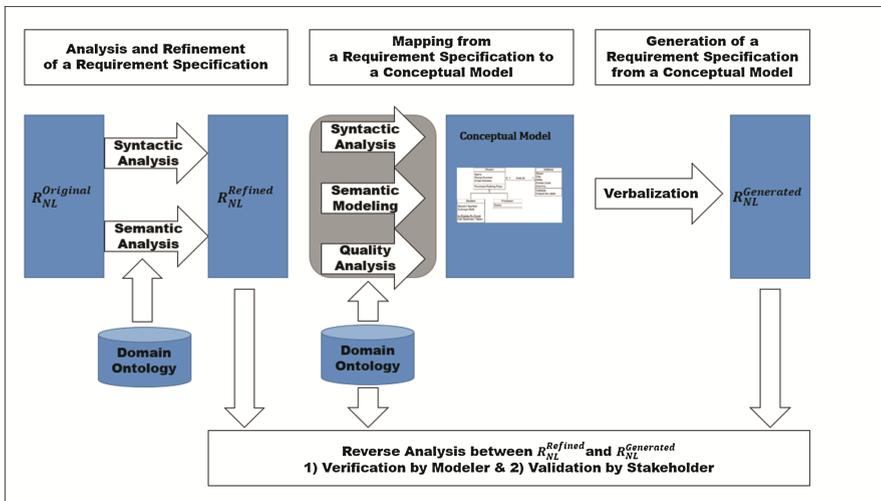

Fig. 1. A framework for semi-automated conceptual modeling

4.1   Analysis and Refinement of a Requirement Specification

In the first step, the syntax and semantics of the requirement specification in NL is analyzed and refined. The requirement specification in NL must be syntactically complete. Most of previous studies simply assume that requirements are complete and free from syntactic errors. Hence, it is necessary to check the syntax of the requirement specification before it is fed into an automated tool. An incomplete requirement specification is treated to have syntactic errors, and the modeler should be informed so that the errors could be corrected.

In order to check semantic quality of the specification, a domain ontology related to the specification is most effective [30]. In order to build the domain knowledge and associate it to the requirement specification, the methodology proposed in [31] can be used. A systematic analysis and refinement of a requirement specification using a sophisticated NLP tool that utilizes a domain-specific ontology is necessary. However, the checking of the quality of a specification, in terms of syntactic and semantic completeness and accuracy, needs to be studied more.

4.2   Mapping from a Requirement Specification to a Conceptual Model

In the second step, a conceptual model is generated from the refined requirement specification. The conceptual model should be generated by integrating syntactic analysis, semantic modeling with domain knowledge, and quality analysis. Ideally, multiple methodologies discussed in Sect. 3 should be combined and used.

The syntax and semantics of a requirement specification should be analyzed separately. The syntactic analysis of a requirement specification can be accomplished by NLP tools. The first easy-to-use technique is the noun-identification approach, which evaluates nouns in the requirement specification to find classes or entities in the conceptual model [18, 51]. The patterns of specific sentences [41] or verbs [54] are retrieved to find appropriate relationships. Since these techniques are not complete, accurate or even generic, we need to use other techniques to identify additional entities that cannot be discovered by pure syntax analysis. We can use the taxonomic class modeling methods [59] to identify entities that can be inferred from verbs and entities that are important to the domain but are not explicitly stated in the specification. Song et al. use rule-based methods to identify entities from verbs and apply domain knowledge to the notion of entity categories to discover those implicit entities [59].

Based on the results of the syntactic analysis, the semantic analysis of the requirement specification should be followed. A popular way to add semantics during the automated generation of a model is to exploit a lexical database such as WordNet as well as a domain ontology. Additional semantic modeling approaches that can be employed are Discourse model [32], Interlingua [38], ROM [69], and SemNet [43]. For more meaningful semantic modeling, techniques of utilizing the domain knowledge with these semantic modeling approaches should be studied.

Next, the ways of enforcing constraints of a domain or a modeling language and the quality of final conceptual models should be considered. A simple way to effectively enforce constraints is to employ a metamodel of the target conceptual model. Constraint

types specified in the metamodel help the modeler identify and define domain-specific constraints. An automated quality analysis of conceptual models is challenging. Aguilera et al. [2] present a method that defines quality issues of a given model and detects quality issues of the model. Another approach is to define domain-specific quality issues by developing a UML profile [27]. An automated mapping methodology that incorporates the domain-specific quality issues needs much more investigation. To realize the automation of the aforementioned processes, a series of steps such as syntax analysis, semantic modeling, domain knowledge, and constraints and quality enforcement must be systematically integrated.

4.3 Reverse Analysis Between the Refined Requirement Specification and the Generated Requirement Specification

Evaluation methods are required to examine whether the generated conceptual data model represents the necessary and sufficient conditions of the semantics of the requirement specification. Ideally, both the data modeler and stakeholders (or domain experts) should participate in the evaluation process. The data modeler can verify the syntax and semantics of the model against the requirement specification as he/she is well versed with the employed data model. Stakeholders can validate the semantics of the generated model as they know more about the domain than the data modeler does. In order to help the stakeholders validate the model against the requirement specification, the generated conceptual model should be translated into a description in NL that could be easily understood by the stakeholders. The semantics of generated description can be evaluated against the original requirement specification and the domain. A verbalization method, used to produce a verbalization of a UML conceptual schema [1], may be used for this purpose.

In previous studies, the conceptual models are summarized in terms of recall and precision based on the experts' analyses [35]. Comparative analyses between the conceptual model in NL and the requirement specification are also necessary. For this purpose, both the comparison aspect of semantics at the sentence level and the development of quantitative measurements are also required. Semantic recall and precision [10] could be employed for that purpose.

4.4 Needs for Rational Empirical Studies

For effective comparisons of various conceptual modeling techniques and tools, it is necessary to have consistent, reliable, and replicable testbed and benchmark tests. Our survey shows that there have been no such environments in the conceptual modeling community. A lack of these standard testing environments limits comparative studies.

These testing environments should include multiple specific application areas with their requirement specifications; domain knowledge of the areas; automated conceptual modeling methodologies and tools to create conceptual models; methods of translating the generated conceptual model into diverse natural languages; and quantitative measures for comparing the results of the testing. In order to create these environments, contributions by crowdsourcing would be desirable.

5   Conclusions

This paper presented six categories of methodologies that can be used for semi-auto-mated conceptual data modeling. These categories include linguistic-based, pattern-based, case-based, ontology-based, metamodeling-based, and multi-techniques-based conceptual modeling. Based on this survey, we proposed a three-step framework for semi-automated conceptual data modeling. It consists of an analysis and refinement of the requirement specification, a mapping of the requirement specification to a conceptual model, and a reverse analysis between the generated requirement specification from the generated conceptual model and the input requirement specification. We proposed the need for testbeds and benchmark testing environments for conceptual modeling research. We also discussed promising research topics in each step. Automated concep-tual data modeling is far from satisfactory and there are many challenging issues ahead. Recent advances in cognitive computing [46] and machine learning [68] techniques could be introduced in automated conceptual modeling.

References

1. Aguilera, D., García-Ranea, R., Gómez, C., Olivé, A.: An eclipse plugin for validating names in UML conceptual schemas. In: De Troyer, O., Medeiros, C.B., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H. (eds.) ER 2011. LNCS, vol. 6999, pp. 323–327. Springer, Heidelberg (2011)
2. Aguilera, D., Gómez, C., Olivé, A.: A method for the definition and treatment of conceptual schema quality issues. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 501–514. Springer, Heidelberg (2012)
3. Ambriola, V., Gervasi, V.: On the systematic analysis of natural language requirements with circe. Autom. Softw. Eng. **13**, 107–167 (2006)
4. Alexander, C.: The Timeless Way of Building. Oxford University Press, New York (1979)
5. Anthony, S., Mellarkod, V.: Data modeling patterns: a method and evaluation. In: Proceedings of the Fifteenth Americas Conference on Information Systems, San Francisco, California (2009)
6. Atkinson, C., Kuhne, T.: Model-driven development: a metamodeling foundation. IEEE Softw. **20**(5), 36–41 (2003)
7. Batra, D.: Cognitive complexity in data modeling: causes and recommendations. Requirements Eng. **12**(4), 231–244 (2007)
8. Blaha, M.: Patterns of Data Modeling. CRC Press, Boca Raton (2010)
9. Buchholz, E., Cyriaks, H., Dsterhft, A., Mehlan, H., Thalheim, B.: Applying a natural language dialogue tool for designing databases. In: Proceedings of the first International Workshop on Applications of Natural Language to Databases (NLDB 1995) (1995)
10. Burton-Jones, A., Meso, P.: How good are these UML diagrams? An empirical test of the Wand and Weber good decomposition model. In: ICIS 2002 Proceedings, 10 (2002)
11. Chaiyasut, P., Shanks, G.: Conceptual data modeling process: a study of novice and expert data modelers. In: Proceedings of the 1st International Conference on Object-Role Modeling, Australia, University of Queensland (1994)
12. Chen, P.: English sentence structure and entity-relationship diagram. Inf. Sci. **1**(1), 127–149 (1983)

13. Chen, P.: The entity-relationship model: toward a unified view of data. ACM Trans. Database Syst. **1**(1), 9–36 (1976)
14. Choobineh, J., Lo, A.: CABSYDD: case-based system for database design. J. Manag. Inf. Syst. **21**(3), 242–253 (2004)
15. Coad, P., North, D., Mayfield, M.: Object Models – Strategies, Pattern, and Applications. Yourdon Press, Englewood Cliffs (1995)
16. Conesa, J., Storey, V.C., Sugumaran, V.: Experiences using the ResearchCyc upper level ontology. In: Kedad, Z., Lammari, N., Métais, E., Meziane, F., Rezgui, Y. (eds.) NLDB 2007. LNCS, vol. 4592, pp. 143–155. Springer, Heidelberg (2007)
17. Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A.: Methodologies, tools and languages for building ontologies: where is their meeting point? Data Knowl. Eng. **46**, 41–64 (2003)
18. Deeptimahanti, D.K, Sanyal, R.: Semi-automatic generation of UML models from natural language requirements. In: Proceedings of the 4th India Software Engineering Conference (ISEC 2011), pp. 165–174 (2011)
19. Dehne, F., Steuten, A., van de Riet, R.P.: WordNet++: a lexicon for the color-X method. Data Knowl. Eng. **38**(1), 3–29 (2001)
20. El-Ghalayini, H., Odeh, M., McClatchey, R.: Engineering conceptual data models from domain ontologies: a critical evaluation. Int. J. Inf. Technol. Web Eng. **2**(1), 57–70 (2006)
21. Embley, D.: Toward semantic understanding: an approach based on information extraction ontologies. In: Proceedings of the 15th Australian Database Conference, Denedin, New Zealand, pp. 3–12 (2004)
22. Evermann, J., Wand, Y.: Towards ontologically-based semantics for UML constructs. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 354–367. Springer, Heidelberg (2001)
23. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
24. Fill, H.-G., Karagiannis, D.: On the conceptualization of modelling methods using the ADOxx meta modeling platform. Enterp. Model. Inf. Syst. Archit. **8**(1), 4–25 (2013)
25. Fonseca, F., Martin, J.: Learning the differences between ontologies and conceptual schemas through ontology-driven information systems. JAIS – J. Assoc. Inf. Syst. Spec. Issue Ontol. Context IS **8**(2), 129–142 (2007)
26. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison Wesley, Menlo Park (1997)
27. Gailly F., Poels, G.: Conceptual modeling using domain ontologies: improving the domain-specific quality of conceptual schemas. In: Proceedings of the 10th Workshop on Domain-Specific Modeling, pp. 18–24 (2010)
28. Gogolla, M., Hohenstein, U.: Towards a semantic view of an extended entity-relationship model. ACM Trans. Database Syst. **16**(3), 369–416 (1991)
29. Conesa, J., Storey, V., Sugumaran, V.: Usability of upper level ontologies: the case of ResearchCyc. Data Knowl. Eng. **69**(4) (2010)
30. Gnesi, S., Fabbrini, F., Fusani, M., Trentanni, G.: An automatic tool for the analysis of natural language requirements. informe técnico, CNR Information Science and Technology Institute, pp. 53–62 (2004)
31. Han, T., Purao, S., Storey, V.: Generating large-scale repositories of reusable artifacts for conceptual design of information systems. Decis. Support Syst. **45**, 665–680 (2008)
32. Harmain, M., Gaizauskas, R.: CM-builder: a natural language-based CASE tool for OO analysis. J. Autom. Softw. Eng. **10**(2), 157–181 (2003)
33. Hartmann, S., Link, S.: English sentence structures and EER modeling. In: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modeling (2007)
34. Hay, D.C.: Data Model Patterns: Conventions of Thought. Dorset House Publishing, New York (1996)

35. Jarvenpaa, S.L., Machesky, J.J.: Data analysis and learning: an experimental study of data modeling tools. Int. J. Man Mach. Stud. **31**(4), 367–391 (1989)
36. Kim, N., Lee, S., Moon, S.: Formalized entity extraction methodology for changeable business requirements. J. Inf. Sci. Eng. **24**, 649–671 (2008)
37. Kim, Y., March, S.: Comparing data modeling formalisms. Commun. ACM **38**(6), 103–115 (1995)
38. Kop, C., Fliedl, G. Mayr, H.: From natural language requirements to a conceptual model. In: Proceeding of the First International Workshop on Evolution Support for Model-Based Development and Testing (EMDT2010), pp. 67–73 (2010)
39. Lenat, D.B.: CYC: a large-scale investment in knowledge infrastructure. Commun. ACM **38**(11), 33–38 (1995)
40. Luisa, M., Mariangela, F., Pierluigi, N.I.: Market research for requirements analysis using linguistic tools. Requirements Eng. **9**(1), 40–56 (2004)
41. Mala, A., Uma, V.: Automatic construction of object oriented design models [UML diagrams] from natural language requirements specification. In: Proceedings of the 9th Pacific Rim international conference on Artificial intelligence (PRICAI 2006), pp. 1155–1159 (2006)
42. Mascardi, V., Cordì, V., Rosso, P.: A comparison of upper ontologies. Technical report DISI-TR-06-2 (2007)
43. Mich, L., Garigliano, R.: The NL-OOPS project: object oriented modeling using the natural language processing system LOLITA. In: Proceedings of the 4th International Conference on the Applications of Natural Language to Information Systems (NLDB 1999) (1999)
44. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
45. Miyoshi, H., Sugiyama, K., Kobayashi, M., Ogino, T.: An overview of the EDR electronic dictionary and the current status of its utilization. In: Proceedings of the 16th International Conference on Computational Linguistics (1996)
46. Modha, D.S., Ananthanarayanan, R., Esser, S.K., Ndirango, A., Sherbondy, A.J., Singh, R.: Cognitive computing. Commun. ACM **54**(8), 62–71 (2011)
47. Moody, D.L.: Metrics for evaluating the quality of entity relationship models. In: Ling, T.W., Ram, S., Lee, M.L. (eds.) ER 1998. LNCS, vol. 1507, pp. 211–225. Springer, Heidelberg (1998)
48. Moody, D.L., Shanks, G.G.: What makes a good data model? Evaluating the quality of entity relationship models. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 94–111. Springer, Heidelberg (1994)
49. Moody, D.L., Shanks, G.G.: Improving the quality of data models: empirical validation of a quality management framework. Inf. Syst. **28**(6), 619–650 (2003)
50. Neill, C., Laplante, P.: Requirement engineering: the state of the practice. IEEE Softw. **20**(6), 40–45 (2003)
51. Omar, N,. Hanna, P., Mc Kevitt, P.: Heuristics-based entity-relationship modelling through natural language processing. In: Proceedings of the Fifteenth Irish Conference on Artificial Intelligence and Cognitive Science (AICS-04), pp. 302–313 (2004)
52. Paek, Y.K., Seo, J., Kim, G.C.: An expert system with case-based reasoning for database schema design. Decis. Support Syst. **18**(1), 83–95 (1996)
53. Parson, J., Saunders, C.: Cognitive heuristics in software engineering: applying and extending anchoring and adjustment to artifact reuse. IEEE Trans. Softw. Eng. **30**(12), 873–888 (2004)
54. Popescu, D., Rugaber, S., Medvidovic, N., Berry, D.M.: Reducing ambiguities in requirements specifications via automatically created object-oriented models. In: Martell, C. (ed.) Monterey Workshop 2007. LNCS, vol. 5320, pp. 103–124. Springer, Heidelberg (2008)
55. Purao, S., Storey, V.C.: A multi-layered ontology for comparing relationship semantics in conceptual models of databases. J. Appl. Ontol. **1**(1), 117–139 (2005)

56. Simsion, G.: Data Modeling Theory and Practice. Technique Publications, LLC, New York (2007)
57. Soares, A., Fonseca, F.: Ontology-Driven Information Systems at Development Time. IJCSS – J. Comput. Syst. Signals **8**(2) (2007)
58. Song, I.-Y., Evans, M., Park, E.: A comparative analysis of entity-relationship diagrams. J. Comput. Softw. Eng. **3**(4), 427–459 (1995)
59. 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, Hershey (2004)
60. Sprinkle, J., Rumpe, B., Vangheluwe, H., Karsai, G.: Metamodeling - state of the art and research challenges. In: Giese, H., Karsai, G., Lee, E., Rumpe, B., Schätz, B. (eds.) MBEERTS. LNCS, vol. 6100, pp. 57–76. Springer, Heidelberg (2008)
61. Storey, V.C.: Classifying and comparing relationships in conceptual modeling. IEEE Trans. Knowl. Data Eng. **17**(11), 1–13 (2005)
62. Storey, V.C.: Understanding semantic relationships. VLDB J. **2**, 455–488 (1993)
63. Storey, V.C., Chiang, R., Goldstein, R., Dey, D., Sundaresan, S.: Database design with common sense business reasoning and learning. ACM Trans. Database Syst. **22**(4), 471–512 (1997)
64. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer, Berlin (2000)
65. Thonggoom, O., Song, I.-Y., An, Y.: EIPW: a knowledge-based database modeling tool. In: Salinesi, C., Pastor, O. (eds.) CAiSE Workshops 2011. LNBIP, vol. 83, pp. 119–133. Springer, Heidelberg (2011)
66. Thonggoom, O., Song, I.-Y., An, Y.: Semi-automatic conceptual data modeling using entity and relationship instance repositories. In: Jeusfeld, M., Delcambre, L., Ling, T.-W. (eds.) ER 2011. LNCS, vol. 6998, pp. 219–232. Springer, Heidelberg (2011)
67. Topi, H., Ramesh, V.: Human factors research on data modeling: a review of prior research, an extended framework and future research directions. J. Database Manag. **13**, 3–15 (2002)
68. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, Los Altos (2005)
69. Zeng, Y.: Recursive object model (ROM)-modeling of linguistic information in engineering design. J. Comput. Ind. **59**, 612–625 (2008)

Springer