A Semi-Automatic Method for Matching Schema Elements in the Integration of Structural Pre-Design Schemata

Peter Bellström
Department of Information Systems
Karlstad University
Karlstad, Sweden
Peter.Bellstrom@kau.se

Jürgen Vöhringer
econob GmbH
Klagenfurt, Austria
juergen.voehringer@econob.com

***Abstract*—In this paper we present a semi-automatic method for matching schema elements in the integration of structural pre-design schemata. In doing so we describe and present how element level matching (concept), structural level matching (neighborhood) and taxonomy-based matching can be combined into one workflow and method. The matching method is a composite schema-based matching method where several different approaches are used to receive one single matching result. Our contributions facilitate the otherwise complex task of matching schema elements during the integration of pre-design schemata and they also speed up the process due to automation of certain process steps. The research approach used within this work can be characterized as design science and our main contributions as a *method* and an *instantiation* (a prototype).**

Keywords-Semi-automaic Schema Integration; Pre-Design; Schema Matching; Implementation Neutral Design

I. INTRODUCTION

In the early phases of information system development we deal with requirements that are described in natural language and suitable modeling languages, resulting in a number of documents and schemata. These schemata both illustrate structural (static) and behavioral (dynamic) aspects. The requirements, however, are not illustrated in one schema but in a set of schemata, each showing some small fraction of the information system being designed. To avoid problems and misunderstandings these schemata should be integrated into one blueprint of the information system. In other words, the source schemata are to be integrated into one global conceptual schema. The schema integration process is divided into at least four phases: it starts with a *preparation* phase, then moves on to a *comparison* phase, which is followed by a *resolution* phase and ends with a phase in which the schemata are *superimposed* and the global integrated schema is *restructured*. The focus of this paper is on the second of theses phases; i.e. how to recognize similarities and differences between the compared source schemata. In [1] we described a three-tier matching strategy for pre-design schema elements that facilitates the difficult task of pair-wise comparison of the source schemata while aiming to recognize similarities and differences between them. In this paper we present and describe a continuation and extension of that work. More precisely, this means that we present and describe a semi-automatic method for matching schema elements during the integration of structural pre-design schemata. The recognition of similarities and differences is one of the integration phases that can be automated, which is something we should aim for, because, as Doan et al. [2] express it, “schema matching today remains a very difficult problem.” (p. 11).

One of the most quoted descriptions of ‘schema integration’ is given by Batini et al. [3], who state that schema integration is “the activity of integrating the schemas of existing or proposed databases into a global, unified schema.” (p. 323). Since schema integration is a very complex, error-prone and time-consuming task [4], computer-based applications and tools are needed to facilitate the process. Consequently, in this paper we present a semi-automatic method for matching schema elements in the integration of structural pre-design schemata. In this method focus is placed on automation with the main goal to consolidate different matching strategies and approaches in order to achieve semi-automatic recognition of similarities and differences between schemata. By it we always compare two source schemata since binary ladder integration is assumed [3][5]. Even though automatic schema integration is desirable, we agree with Stumptner et al. [6] who state in connection with dynamic schema integration, full automation is not feasible due to the complexity of the task. We also argue that domain experts are an important source of domain knowledge and therefore should be involved in the entire schema integration process. In this article the compared structural schemata only contain two types of primitives: concepts (including labels) and connections (dependency/relationship) between the concepts. Finally, our use of ‘pre-design’ refers to analysis and design on an implementation-independent level; i.e. focusing on describing the content (what) rather than the specific implementations of an information system (how). Besides schema integration, another application area for pre-design matching is the consolidation of project schemata during ontology engineering (see for instance [7]).

The article is structured as follows: in section two we describe the applied research approach. In section three we address related work and distinguish it from our own. In section four we present the schema integration process and in section five this article’s main contribution is discussed: the proposed semi-automatic schema matching method. Finally, the paper closes with a summary and conclusions.

II. Research Approach

The research approach used within this work can be characterized as design science. The big difference between the 'behavioral science paradigm' and the 'design science paradigm' is that behavior science "seeks to find 'what is true'" while design science "seeks to create 'what is effective'" [8](p. 98). Design science in not a new research approach. It has been used for a rather long time in several disciplines such as Computer Science, Software Engineering and Information Systems [9]. In design science focus is on producing artifacts. Hevner et al. [8] describe it as follows: "The result of design-science research in IS is, by definition, a purposeful IT artifact created to address an important organizational problem." (p. 82). In this quotation, Hevner et al. use IS as an acronym for 'Information System.' The produced artifacts can further be classified as *constructs*, *models*, *methods* and *instantiations* [8][10]. In Table 1, each type of artifact is briefly described quoting March & Smith [10].

TABLE I. Design Science Artifacts according to [36]

ARTIFACT	DESCRIPTION
Construct	*Construct* or concepts form the vocabulary of a domain. They constitute a conceptualization used to describe problems within the domain and to specify their solutions. (p. 256)
Model	A *model* is a set of propositions or statements expressing relationships among constructs. In design activities, models represent situations as problem and solution statements. (p. 256)
Method	A *method* is a set of steps (an algorithm or guideline) used to perform a task. (p. 257)
Instantiation	An *instantiation* is the realization of an artifact in its environment. (p. 258)

Finally, it is important to evaluate design science research contributions through one, or several, evaluation methods. In [8] Hevner et al. describe five such evaluation methods: *observational*, *analytical*, *experimental*, *testing* and *descriptive*. In Table 2, each evaluation method is shortly described quoting [8].

As will be addressed in section 5, our research has two types of contributions: we have developed a *method* and an *instantiation* (a prototype). To validate these research contributions several evaluation methods have been used: *analytical*, *testing* and *descriptive* evaluation methods.

For a more detailed discussion and description of the design science approach, please see Hevner & Chatterjee [11].

TABLE II. Design Science Evaluation Methods according to [27]

EVALUATION METHOD	DESCRIPTION
1. Observational	*Case Study:* Study artifact in depth in business environment *Field Study:* Monitor use of artifact in multiple projects (p. 86)
2. Analytical	*Static Analysis:* Examine structure of artifact for static qualities (e.g., complexity) *Architecture Analysis:* Study fit of artifact into technical IS architecture *Optimization:* Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior *Dynamic Analysis:* Study artifact in use for dynamic qualities (e.g., performance) (p. 86)
3. Experimental	*Controlled Experiment:* Study artifact in controlled environment for qualities (e.g., usability) *Simulation* - Execute artifact with artificial data (p. 86)
4. Testing	*Functional (Black Box) Testing:* Execute artifact interfaces to discover failures and identify defects *Structural (White Box) Testing:* Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation (p. 86)
5. Descriptive	*Informed Argument:* Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility *Scenarios:* Construct detailed scenarios around the artifact to demonstrate its utility (p. 86)

III. Previous and Related Work

In the schema integration research field several approaches and methods have been proposed during the last thirty years. These can roughly be classified into three approaches (see Bellström [12]): manual, formal and semi-automatic. *Manual* means that everything is done by hand, *formal* means that a formal modeling language is used and *semi-automatic* means that at least one computer-based tool (application) is used to support the manual steps in the integration process. Our research is mainly placed within semi-automatic, the last of these three approaches. Previously we have both developed automation rules (see [13][14]) and implemented a prototype (see [15][16]). Besides that, we have also consolidated several matching strategies into one applicable matching approach for pre-design schema elements (see [1]). Our research focuses on developing a modeling language-independent integration method. Some preliminary results were given [17] in which we proposed six generic integration guidelines:

- Performing schema integration on the pre-design level
- Standardizing concept notions and utilizing them during integration
- Using domain repositories for supporting the integration process
- Neighborhood-based conflict recognition
- Pattern-based resolution of integration conflicts

- Computer supported integration with utilized user feedback

This was followed by a proposal of a method for modeling language-independent integration of dynamic schemata (see [18]). Here we only focused on modeling language-independent constructs. This means that the focus was only on two primitives – *processes* and *conditions* – with the proposed method being comprised of four phases:

- Preparation of the source schemata
- Recognition of conflicts and commonalties between the source schemata
- Resolution of conflicts and commonalties between the source schemata
- Merging the source schemata and restructuring the global schema

In [18] we also mapped the generic integration strategies proposed in [17] to these aforementioned phases.

As mentioned in the introduction, our research focused on the implementation-neutral level; i.e. implementation-neutral schemata (pre-design schemata) and modeling-independent schema integration. In doing so the numbers of conflicts that can arise are reduced since fewer modeling concepts are needed [19]. Some specific 'pre-design' (user-near) modeling languages do exist today; e.g. Klagenfurt PreDesign Model (KCPM) [20] and Enterprise Modeling (EM) [21]. It is still possible, however, to use any modeling language for pre-design [17].

Looking at previous work in relation to traditional modeling languages it can be concluded that the Entity-Relationship Modeling Language (ERML) [22], or some extension of it, has dominated schema integration research [23] for a long time., Focus, lately, has shifted towards the Unified Modeling Language (UML) [24]. Both the ERML and the UML are modeling languages that are used to illustrate *implementation-dependent* aspects of an information system and are therefore not ideal for our research where we focus on the *implementation-independent* level; i.e. implementation-neutral design. This means that we do not distinguish between entities (classes) and attributes (see for instance [25] and [19]), which is something ERML and UML do. Nor are we interested in implementation-specific features, such as lists, that can be found in both UML and in many programming languages. Instead, we are interested in 'what' and therefore focus on content.

Having addressed previous and related work in relation to traditional modeling languages we now turn to related work in relation to semi-automatic approaches and methods. By doing so, we address some semi-automatic approaches and distinguish them from our own efforts.

Rahm & Bernstein [26] presents a survey on automatic schema-matching approaches. They differentiate *schema-based* and *instance-based* matching. Schema-based matching can be performed on the *element level* (concept) and on the *structural level* (neighborhood). Moreover, the matching can be *linguistic-based* (depending on the similarity of names or descriptions) or *constraint-based* (depending on meta-information about concepts, such as data types and cardinalities). Furthermore, Rahm & Bernstein [26] classify combined approaches as either *hybrid* or *composite* matchers. The difference between these two is that hybrid matchers use different matching approaches independently, whereas composite matchers use different approaches to receive one single result.

Following the terminology of Rahm & Bernstein [26], our own matching method can be classified as a *composite schema-based matching approach* because we apply element-level matching, structural-level matching and taxonomy-based matching with the goal of receiving one single result. Later on the results presented in [26] were adapted, refined and modified by Shvaiko & Euzenat [27].

In Lee & Ling [28] and He & Ling [29], the authors present algorithms for resolving different structural conflicts. These are the conflicts between entity types and attributes, as well as schematic discrepancy. He & Ling [29] express schematic discrepancy as follows: "the same information is modeled as data in one database, but metadata in another." (p. 245). In both [28] and [29] the authors work towards an (semi-) automatic method in which structural conflicts and schematic discrepancy are both resolved by transforming attributes and metadata to entity types.

Our method does not distinguish between entity types (classes) and attributes (properties) because our focus is on implementation-neutral design.

Several algorithms for calculating concept similarity have also been proposed. The *Wu and Palmer* [30] similarity value is one such algorithm, which is calculated using formula 1:

$$wup = \frac{2 * depth(LCS(concept1, concept2))}{depth(concept1) + depth(concept2)} \qquad (1)$$

In a first step the so called LCS (the **L**east **C**ommon **S**ubsumer) is determined; i.e. the first common parent of the compared concepts in the taxonomy. The Wu and Palmer similarity score is then derived from dividing the double of the taxonomy depth of the LCS by the sum of the taxonomy depths of the compared concepts. Further separation of the concepts from their first common father concept means a lower similarity score.

The *Hirst and St Onge* [31] similarity value allows us to measure the similarity between two concepts by determining the length of the taxonomy path between them. The paths for connecting concepts can be distinguished based on their strength: extra-strong, strong and medium paths. *Extra-strong paths* exist between two equivalent concepts; *strong paths* are identified by a direct connection between two concepts while *medium-strong paths* finally mean that two concepts are indirectly connected. In the latter case the number of path direction changes is relevant for determining the concept similarity. Direction changes occur every time a medium-strong connection switches between upwards-paths, downward-paths and horizontal paths. More precisely, this means generalization, specialization and other relationships exist between the concepts. Frequent direction changes lower the similarity score as shown by formula 2:

$$hso = C - pathLength - k * numberDirectionChanges \qquad (2)$$

The calculation returns the value zero if no path at all exists between the concepts. In that case, the concepts are interpreted as unrelated. **C** and **k** are constants used for scaling the similarity value.

Finally, the Lesk similarity value [32] is a context-based similarity score that does not require taxonomic structures. Instead it presupposes a lexicon in which different word senses are distinguished and detailed definitions for each meaning are available. Because the WordNet [33] taxonomy is freely available and contains definitions and examples for each concept, it is a popular choice for this task. For determining the Lesk similarity score the definitions of both involved concepts must be provided so that a numerical estimation of their degree of separation can then be calculated by counting the word overlap.

Some techniques of our schema element-matching method are similar to the ones used in the DIKE [34] and the GeRoMeSuite [35] approaches, but both of these differ to our own method in some key aspects. In contrast to the DIKE approach we do not focus on any specific modeling language nor do we focus on implementation-dependent models, such as SQL, XML or OWL, which the GeRoMeSuite does. It can therefore be concluded that our method and the DIKE and GeRoMeSuite approaches are complementary rather than exclusive.

IV. THE SCHEMA INTEGRATION PROCESS

Since schema integration is a very time consuming and error-prone process it needs to be divided into a number of phases (e.g. [3] [36]). Besides that, it is important that each phase also has a clear purpose and resolves the problems it is set up to deal with. In the rest of this section we present the integration process as stated by Batini et al. [3]. There they point out that the integration process – in this case integration of structural schemata – is composed of the following four phases (see Figure 1):

- Pre-Integration (A)
- Comparison of the schemata (B)
- Conforming the schemata (C)
- Merging and restructuring (D)

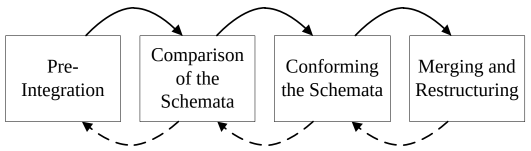

Figure 1.The Schema Integration Process adaped and modified from [7] (p. 20)

The arrows moving from left to right should be interpreted as feed-forward while the arrows from right to left are to be read as feed-back. Figure 1 also illustrates that the schema integration process is highly iterative and that it is possible to move back and forth between the phases. In other words, it is possible to go back to an earlier phase, make adjustments and then move forward again in the integration process.

The phases proposed by Batini et al. [3] have influenced many integration strategies (e.g. [18]) and have been used as the basis for integration methods completely (e.g. [37] [38]) or in part (e.g. [39] [28]). Extensions of the integration process, with an additional phase in between comparing and conforming the schemata, have also been suggested by Bellström [40]. For a more detailed discussion and description of the integration process, please see [12].

A. *Pre-Integration*

The first phase in the schema integration process is *pre-integration*. According to Song [23], this phase includes at least three sub-tasks: (1) translating all schemata into the chosen modeling language (e.g. KCPM or EM), (2) checking for similarities and differences in each schema (e.g. homonyms) and (3), selecting integration strategies (e.g. binary or n-ary integration).

B. *Comparison of the schemata*

The second phase in the schema integration process is *comparison of the schemata*. According to Johannesson [41], this phase also includes at least three sub-tasks: (1) recognition of name conflicts (e.g. synonyms and homonyms), (2) recognition of structural conflicts (e.g. when using ERML one concept is described as an entity type in one schema and as an attribute in another) and (3), recognition of inter-schema properties (e.g. hypernyms-hyponyms and holonyms-meronyms). Schema comparison has been mentioned as an important (see [23]) and difficult (see [42] and [28]) phase of schema integration. During the comparison of schemata, several tasks can be automated successfully.

C. *Conforming the schemata*

The third phase in the schema integration process is *conforming the schemata*. The main task of this phase is to resolve the conflicts and inter-schema properties recognized in the former phase. This phase has also been mentioned as the most critical one (see [28]) and the key issue (see [39]) in schema integration. It should be noted that, during the resolution of conflicts and inter-schema properties, no concepts and dependencies that are important for domain experts must be lost. Losing concepts or dependencies causes semantic loss [43], a problem that in the long run leads to interpretation problems in the integration process.

D. *Merging and restructuring*

The fourth and last phase in the schema integration process is *merging and restructuring*. This phase includes at least two tasks: (1) merging the schemata into one schema and (2), restructuring the integrated schema with the aim to remove redundancy. It is not, however, always clear which dependencies are redundant; i.e. can they be derived from other dependencies as dependencies might have different meanings for different domain experts [44]. Therefore, whenever a slight uncertainty exists whether a concept and/or dependency is redundant, it should be kept in the integrated schema since removing a not truly redundant concept and/or dependency would cause semantic loss [43]. In relation to schema restructuring, the integrated schema should also be checked against several schema qualities such as completeness, minimality and understandability [3] [5].

V. A SEMI-AUTOMATIC METHOD FOR MATCHING PRE-DESIGN SCHEMA ELEMENTS

As indicated in the former section, the schema integration process includes several phases starting with schema preprocessing (pre-integration) moving on to schema matching (comparison and conforming the schemata) and finally ending with schema consolidation (merging and restructuring) (cf. with section IV).

In this article, we view schema preprocessing as a phase in which six activities are carried out:

- Translate the schemata into the chosen modeling language
- Schema element name adaption
- Schema element disambiguation
- Recognition and resolution of inner-schema conflicts
- Introduction of missing relationships
- Selecting the integration strategy

Translate the schemata into the chosen modeling language is the first activity to perform within the preprocessing phase. This activity is applicable when the information system is modeled using several modeling languages. It is, however, very important that the chosen modeling languages “have an expressiveness which is equal to or greater than that of any of the native models” [41](p. 15). For integration of structural pre-design schemata this should not be a problem since concepts and connections between concepts are the only modeling constituents that are used.

Schema element name adaption is the second activity to perform within schema preprocessing. It is a mandatory activity at least in its basic form; i.e. the reduction of words to their base forms using stemmers or lemmatizers. The use of naming guidelines (e.g. [17]) for its modeling elements can further improve a schema. This step is optional because it presupposes an individual set of naming guidelines, which depend at least on the used language.

Schema element disambiguation is an optional, but recommended, step. It constitutes an easy way of improving schema matching and integration on the pre-design level. Word senses are either assigned manually by domain experts or automated suggestions are made based on domain ontologies and general-purpose lexicons.

Recognition and resolution of inner-schema conflicts is mandatory in its basic form; i.e. the same designations are not allowed for different schema elements in static pre-design schemata. It is also recommended to perform an enhanced search for potential homonym and synonym conflicts by using context matching (cf. with section V.B.).

Introduction of missing relationships is optional provided domain ontology, or taxonomy, is available for identifying possible gaps in the schema.

Selecting the integration strategy is the final activity to perform in schema preprocessing and it is a mandatory one. In this activity it must be decided whether binary or n-ary integration should be used [5]. If binary integration is chosen then a further decision must be made whether binary ladder or binary balanced integration (see [3][45]) should be used. For n-ary integration the two options n-ary one-shot, or n-ary iterative (see [3][46]), integration exist. For our method, we have decided to use binary ladder integration.

After schema preprocessing the matching phase starts, which itself consists of several sub phases. We first discussed in detail [25] how the several matching techniques are utilized in a common workflow. An extended and updated version of this process is structured as follows:

Step 1: Preparation for schema element-matching
- *Step 1.1:* Find linguistic base form of schema elements
- *Step 1.2:* Find schema element pairs to be matched

Step 2: Matching on the element level
- *Step 2.1:* Direct element name matching
- *Step 2.2:* Application of linguistic rules
- *Step 2.3:* Domain ontology-based comparison

Step 3: Matching on the structural level
- *Step 3.1:* Determine schema element neighborhood
- *Step 3.2:* Domain ontology-based comparison for all pairs of neighbors
- *Step 3.3:* Rule-based comparison for all pairs of neighbors

Step 4: Taxonomy-based matching

Step 5: Decision on matching results
- *Step 5.1:* Present matching proposals
- *Step 5.2:* Get user feedback
- *Step 5.3:* Finalize matching decisions

In the first schema matching step – *preparation for schema element-matching* – all combinations of schema element pairs from the two source schemata are prepared for comparison. The eventual goal in schema matching is to decide whether an element pair matches. The possible outcomes are:

- Matching
- Related
- Dissimilar

The matching method and workflow are as follows: every schema element pair is first matched on the element level using the direct comparison of the base form and the application of linguistic rules. This step results in a preliminary matching decision. If the result is “dissimilar” and domain ontology is available, then information about potential connections between the elements can be looked up in the ontology. Schema element pairs that are classified as “dissimilar” or “matching” then undergo structural matching, which aims to identify potential conflicts based on the neighbors of the compared elements. If such conflicts are identified, a respective warning is added to the preliminary matching decisions. Finally, taxonomy-based matching – in our case the Lesk algorithm – can be optionally performed for schema element pairs, which are assumed “dissimilar,” in order to detect hidden relatedness. This is especially recommended if at least one of the compared schema elements is as yet unknown in the domain ontology. The final matching proposals, including any warnings, are presented to domain experts who then have the chance to accept the proposals or override them. For instance, they can decide if and how potential differences and similarities, such as homonyms and synonyms, should be resolved. If no domain expert is available the default proposals are pursued. Based on the matching results specific integration proposals are made in the schema consolidation phase. The three different matching steps, which our method is comprised of, are discussed in more detail in the following sections (V.A – V.C).

A. Matching on the element level

A.1 Element name comparison (Step 2.1 and 2.2)

In our matching method element-level matching (concept matching) is the first activity. In element-level matching static schema elements are directly compared via their names (*direct element name matching*). Two elements which have matching names are automatically interpreted as equal for the moment although this decision can be amended later when matching on the structure level is performed. For example, let's assume that schema A and schema B both come from the university domain. The static elements "students" in schema A and "student" in schema B have the same base form: "student." Therefore, the elements match on the element-level; they supposedly describe the same concept.

If one, or both, of the compared schema element names consist of compound words the compounds are deconstructed. For endocentric compounds – the most common ones in the English language according to Bloomfield [47] – the rightmost element of the word is its head. Thus the following two percolative rules (*application of linguistic rules*) are applicable:

a. *If the compared schema elements have names in the form of A and AB (i.e. A corresponds to the compound AB minus the head B), then the relationship* ***"AB belongs/related to A"*** *can be assumed between the elements.*

b. *If the compared schema elements have names in the form B and AB (i.e. A is the head of the compound AB), then the relationship* ***"AB is a B"*** *can be assumed between the elements.*

To exemplify the first rule (a), an element named "car manufacturer" is identified as a potential attribute of an element named "car" ("car manufacturer" belongs to "car") while an element "student social security number" is identified as an attribute candidate for the element "student" ("student social security number" belongs to "student"). However the relationship "belongs to" is usually interpreted as an inverse aggregation, which is not always the intended meaning. For instance "blood pressure" is defined as "the pressure of the circulating blood against the walls of [a person's] blood vessels." Thus it can be preferable to interpret "blood pressure" as an attribute of "person" rather than an attribute of "blood." A more general association named "related to" is preferable in such cases; e.g. blood pressure" related to "blood."

Regarding (b), the second rule, the exemplary element-pair "patient" and "dialysis patient" is interpreted as "dialysis patient" is a "patient," while the concepts "blood pressure measurement" and "measurement" have the relationship "blood pressure measurement" is a (specific form of) "measurement." The "is a" relationship obviously applies to all endocentric compounds, because their head is modified by the rest of the composite (consequently called the modifier) per definition.

A.2 Domain ontology-based comparison (Step 2.3)

The strategy to perform schema element-matching solely based on their names and definitions is not always sufficient. While correlations of element names might indicate a possible matching, the meanings of the involved words might still differ. Moreover, in practice sufficient concept definitions are not always available. Lastly, even if definitions are available for both compared elements they might not be detailed enough to decide whether the schema elements actually match or not. Thus it is optimal to have ontology of the domain at one's disposal.

On the supposition that the compared schema element pair has been preliminarily identified as *dissimilar* in the prior element-level matching step, the matching process proceeds by looking up concepts that fit to the schema element pair in the domain ontology. If both schema elements correspond to concepts in the domain ontology these are again compared on the element-level with the typical outcomes *related* and *dissimilar*. If one of the schema elements is missing no additional information about the elements' relationship can be derived. Nevertheless, the missing elements can still be candidates for new ontology concepts. On the other hand, if both schema elements are found in the domain ontology, then the ontology can be utilized to recognize additional similarity as follows:

- If the elements are directly related by an association in the domain ontology, then their relationship is assumed as "related" and this new relationship is introduced into the integrated schema. If, according to the ontology, both elements are synonymous this is denoted by the special relationship "is synonym of" or "mutual inheritance."

- If the elements are connected indirectly via relationships up to a certain restrictable degree of separation, they may also be assumed as "related." Besides path length, relationship type is also a criterion for whether an indirect connection is interpreted as relatedness. The criteria of path length and relationship type may also be combined when evaluating indirect relationships. In order to correctly depict the connection between the indirectly connected schema elements the missing concepts and relationships from the ontology need to be introduced to the integrated schema.

- If no direct relationship, or indirect path, of the required length and/or type is found in the domain ontology between the compared schema elements, then they are assumed to be dissimilar or independent.

B. Matching on the structural level

B.1 Determine schema element neighborhood (Step 3.1)

The previously described element-based schema matching techniques only take into account the names and definitions of the compared concepts. Matching results obtained in such a way might still be erroneous, however, as both synonym and homonym conflicts are possible (see also Figure 2 and Figure 3).

Matching on the structure level (see [26]) is likely to improve the outcome of element-level matching namely by taking the schema elements' neighborhood into account. Using neighborhood for deriving concept meanings isn't a new approach (e.g. [34]), although it usually is utilized for word-disambiguation in full texts instead of schema element-matching. For instance, Koeling & McCarthy et al. [48] suggest word sense disambiguation based on context and give an example: if the word "plant" has the sense 'industrial plant,' it tends to occur in the neighborhood of words like "factory," "industry," "facility," "business," "company" and "engine." Its other meaning, 'flora,' is often hinted at by neighbors like "tree," "crop," "flower," "leaf," "species," "garden," "field," "seed" and "shrub." Heylen et al. [49] describe similar word matching techniques where 1st and 2nd order bag-of-word-models are used for context-based word matching; i.e. it is determined whether words in the close proximity of the compared target words match. In 1st order models the target words are identified as similar if their neighbors match to a certain degree while in 2nd order models a two-level matching process takes place as the context of the neighbor words themselves are again compared. Context-based word disambiguation techniques, which are applied on natural language texts, usually aim at finding the meaning of single words in the text. Based on context a decision is derived, which out of several possible word definitions is the correct, or most likely, one. This is slightly different from schema matching where word pairs are compared with the goal of identifying possible conflicts. Finding the meaning of each involved concept is the first step towards identifying conflicts and commonalities.

Regarding the notion of neighborhood in natural language text documents, the relevant neighborhood is typically defined by context windows that span a defined number of (content) words before and after the disambiguated word. Additional boundaries are provided by sentence delimiters. When adapting structure-based matching techniques on conceptual schemata, different definitions for the notions of "neighborhood" and "context" are needed. Typically, the context, or neighborhood, of a static schema element encompasses all surrounding directly connected elements. Sometimes it is helpful for the sake of matching and integration algorithms to extend the notion of context so that it also encompasses indirectly connected schema elements up to a certain level of separation.

B.2 Domain Ontology-Based Neighborhood Comparison (Step 3.2)

Essentially neighborhood matching acts as a security check whether the preliminary results from the element level are trustworthy, or if conflicts, such as homonyms or synonyms, are likely. If the neighborhood comparison supports the result of the element-level matching then the integration process continues. But when the neighborhood comparison yields different results than the element-level comparison, however, a contradiction is at hand and the domain experts must be notified by displaying a warning that the current schema element pair has a potential conflict. For the purpose of recognizing conflict candidates two threshold values are defined: the *homonymy threshold* and the *synonymy threshold*. These thresholds are reference values and adaptable so that they can fit the needs of different projects. It is possible for the two thresholds to congregate on the same value, but the homonymy threshold must not be greater than the synonymy threshold.

In order to calculate a matching result that can be compared against the threshold values the structure-level matching of neighbor schema elements is itself performed on the element level. The neighbor elements are counted as matching if one of the following conditions is fulfilled:

- They have the same name and/or definitions
- A synonymy relationship exists between them according to the domain ontology
- The schema element pair has a taxonomy-based similarity score above a certain threshold

Including more than just the direct neighbors in the decision process achieves a more thorough matching, but it is slower due to several more elements to compare. To determine the neighborhood matching of schema element-pairs the percentage of corresponding neighbors is calculated using the matching criteria listed above. For this purpose the following intuitive formula (subsequently referenced as **D**egree of **N**eighborhood **M**atching (DNM)) can be used:

$$\frac{\text{MatchingNeighborSchemaElementCount}}{(\text{NeighborCount}(\text{elementFromSchema1}) + (\text{NeighborCount}(\text{ElementFromSchema2}))} * 1/2 \qquad (3)$$

Formula 3 calculates the percentage of matches among the average neighbor count. The resulting DNM value is compared against the thresholds. Schema elements that have been classified as "matching" on the element level must have a DNM above the defined homonymy threshold. If the DNM is below the threshold a homonym warning is issued by the matching application. Similarly, an element pair that has been classified as "dissimilar" on the element level should be below the provided synonymy threshold. If that isn't the case a synonymy warning is issued by the matching application.

Figures 2 and 3 give examples how analyses of the compared schema elements' context may point towards hidden homonym and synonym conflicts in static pre-design schemata. In Figure 2 the element "course" occurs in both source schemata, but upon nearer inspection of the neighbors it becomes apparent that the two elements have disjunctive neighborhoods.

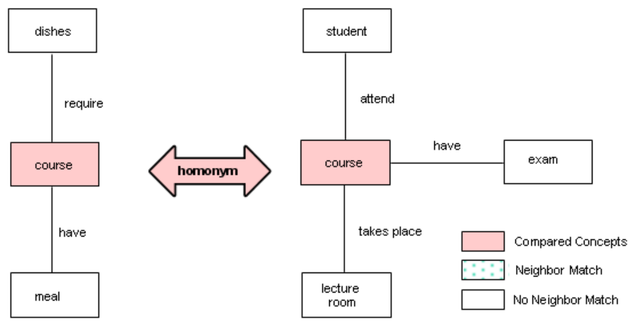

Figure 2. Recognition of homonym conflict [11] (p.113)

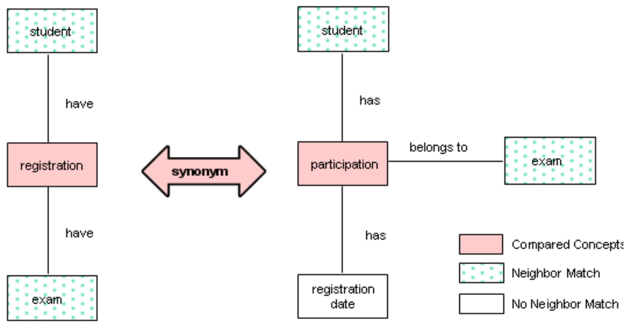

Figure 3. Recognition synonym conflict [11] (p. 114)

This raises the suspicion of homonymy. Indeed, the left schema describes a course served during a meal while the right hand schema describes a university course. It must be noted that severe homonym conflicts as shown in Figure 2 are unlikely to occur if the source schemata come from the same domain. Most remaining homonym conflicts occur between concepts that have less pronounced semantic differences; e.g. see the more extensive example in Figure 4 when the vocabulary of terms is small [3] and when incomplete concept names are used [50].

A synonym conflict is shown in Figure 3. The schema elements "registration" and "participation" have been classified as dissimilar on the element level, but several neighbors match: two of two adjacent elements for the schema element "registration" and two of three neighbors for the schema element "participation." In spite of the small number of neighbors this implies that the elements "participation" and "registration" are used synonymously here. In actuality a student's participation in an exam is modeled in both cases. Synonym conflicts frequently occur when different groups, or departments, are involved who might use different nomenclature for the same concept that's specific to their point of view.

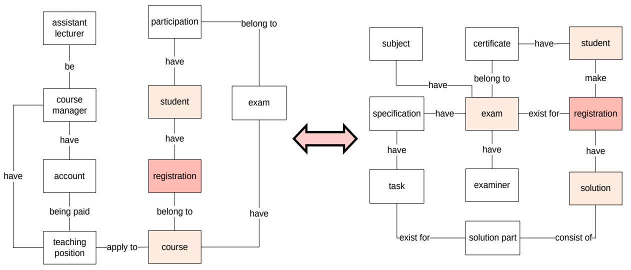

Figure 4. Homonym conflict example [49] (p. 56)

Figure 4 shows an example of a homonym conflict. Since the element "registration" occurs in both source schemata the schema element pair is preliminarily categorized as equivalent on the element level. Comparing the immediate context of the neighborhoods of "registration" the elements show the two registrations as actually being different concepts. Students are involved in both cases, but in the left-hand schema students register to a course, while in the right-hand-schema students register to an exam. In the latter case the students' exam solution is related to their registration too. The result value according to the default DNM formula given above is 0.4, which, depending on the threshold, might be enough to issue a homonym conflict warning by the matching application.

A way of further refining the DNM formula and improving its results is the additional consideration of schema elements that only appear in the neighborhood of one element although they are available in both schemata. Since these elements are expected, but not actual neighbors, in one of the schemata, they could be included in that schema's neighbor count. Naturally, however, the match count stays the same so the resulting DNM score is always lowered when missing expected neighbors are taken into consideration.

B.3 Rule-based Neighborhood Comparison (Step 3.3)

In the third and last activity in structural-level matching we use two types of "IF THEN" rules: those for equivalent element names and others for similar element names (see also [13] and [14]). For this case equivalent means that matching on the element level resulted in two equivalent element names. *Size*, for instance, is an element within both schema 1 and schema 2. On the other hand, similar means that the element names are not equivalent but still have something in common; e.g. *Order Line* in schema 1 and *Order* in schema 2. Our method uses at least six rules for equivalent element names and three rules for similar element names. Our rules for *equivalent element names* can be stated as:

IF the comparison of concept names yields *equivalent* and the comparison of concept neighborhoods yields:

- Equivalent, THEN ***equivalent*** concepts are most likely recognized (E1)
- Different, THEN ***homonyms*** are most likely recognized (E2)
- Similar, AND one concept in each schema is named differently, THEN ***synonyms*** are most likely recognized (E3)

- Similar, AND one concept name is a composite of another concept name with a following addition AND the cardinality is indicating 1:1, THEN an ***association*** between the two concepts is most likely recognized (E4)
- Similar, AND one concept name is a composite of another concept name with a prior addition, THEN a ***hypernym-hyponym pair*** is most likely recognized (E5)
- Similar, AND one concept name is a composite of another concept name with a following addition AND the cardinality is indicating 1:M with or without uniqueness, THEN a ***holonym-meronym pair*** is most likely recognized (E6)

Examples of rules E5 and E6 are illustrated in Figure 5 and Figure 6.

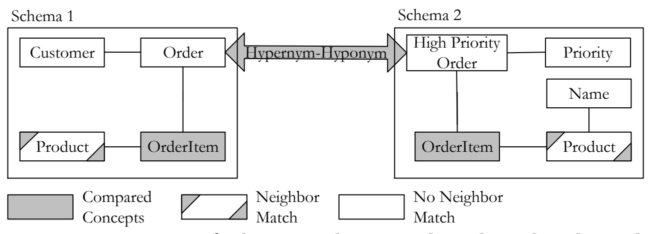

Figure 5. Recognition of a hypernym-hyponym dependency based on rule [10] (p. 184)

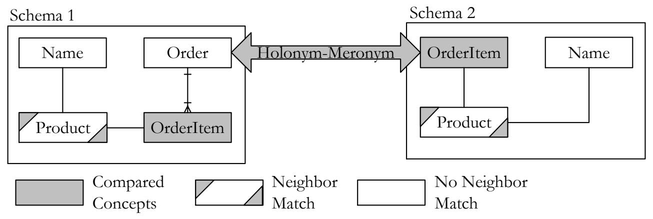

Figure 6. Recognition of a holonym-meronym dependency based on rule [10] (p. 185)

Our rules for *similar concept names* can be stated as:
IF the comparison of concept names yields:

- *Similar*, AND one concept name is a composite of another concept name with a following addition AND the comparison of concept neighborhoods yields similar or equivalent with an indication to a 1:1 cardinality, THEN an ***association*** between the two concepts is most likely recognized
- *Similar*, AND one concept name is a composite of another concept name with a following addition AND the comparison of concept neighborhoods yields similar or equivalent with or without an indication to a unique 1:M cardinality, THEN a ***holonym-meronym pair*** is most likely recognized
- *Similar*, AND one concept name is a composite of another concept name with a prior addition AND the comparison of concept neighborhoods yields similar or equivalent, THEN a ***hypernym-hyponym pair*** is most likely recognized

One last remark is needed before moving on to the Lesk algorithm. The rules addressed above were applied in [13] and [14] while using the Karlstad Enterprise Modeling approach [21]. On the other hand, we do not focus here on any specific modeling language. We have therefore refined and adapted the rules to be useful for any modeling language; in other words, the rules are now modeling language-independent and can be used on the implementation -neutral level.

C. *Taxonymy-based matching - The Lesk Metric (Step 4)*

C.1 The Lesk approach in structure-based matching

The Lesk metric [32] is a domain-independent similarity score that doesn't require taxonomic structures (cf. [30] [31]). Instead, it presupposes a lexicon in which different word senses are distinguished and detailed definitions for each meaning are provided. Because the WordNet [33] taxonomy contains definitions and examples for each concept it is a popular choice for this kind of task. The Lesk approach is a context-based similarity measurement strategy and requires neither the LCS (**L**east **C**ommon **S**ubsumer) nor the path length unlike other WordNet-based similarity measures (cf. [30] [31]). For determining the Lesk similarity score the definitions of both involved concepts must be provided and then a numerical estimation of their degree of separation is calculated by counting the word overlap.

In Banerjee & Pederson et al. [32] and Ekedahl & Golub et al. [51] specific implementations of the Lesk-algorithm are discussed for disambiguating words in full texts using WordNet [33]. In their approach a context window containing an equal number of words on both sides of the observed word is defined after which all available definitions for the observed concept and the other content words in the context window are examined and compared. The word sense that has the *greatest overlap* with the definitions from the surrounding text is assumed to be the correct one. Non-content words, like pronouns or articles, are excluded from the overlap count. Although the Lesk algorithm was originally designed for word disambiguation in full texts a similar approach can be applied to schemata by comparing not only concept notions, but also the definitions of concepts and those of their neighbors. This application of the Lesk algorithm for disambiguation purposes is similar to schema matching on the structure level although concept definitions rather than concept names are matched.

C.2 Calculation and performance of the Lesk score

In our method we adopt the Lesk algorithm as presented by Vöhringer & Fliedlet al. in [15]. The following glosses are interpreted for each compared word: the examples and the definitions of the hypernyms, hyponyms, meronyms, holonyms and the word itself. Table III lists the glosses for the concept "bus#n#1" as extracted from WordNet. The underlined words are not part of the actual gloss value but added for clarity reasons because they denote the corresponding concepts of the gloss.

Permutations of the various glosses are tested for overlaps; i.e. example "car#n#1"- glos "bus#n#1," glos "car#n#1"- glos "bus#n#1" etc. (see table IV). In order to prioritize longer matches, the score for each gloss-pair is defined as the sum of the squared word count of each overlap. The total Lesk similarity score is defined as the accumulated score of all gloss-combinations.

TABLE III. TYPICAL WORDNET GLOSSES FOR THE CONCEPT “BUS#N#1“

Gloss Type	Description	Gloss Value
Example	Example of usage	"he always rode the bus to work"
Glos	Word sense definition	bus: a vehicle carrying many passengers; used for public transport;
Hype glos	Bus is a kind of…	public transport: conveyance for passengers or mail or freight
Hypo glos	… Is a kind of bus	minibus: a light bus (4 to 10 passengers) school bus: a bus used to transport children to or from school trolleybus: a passenger bus with an electric motor that draws power from overhead wires
Holo glos	Bus is a part of…	fleet: group of motor vehicles operating together under the same ownership
Mero glos	… Is a part of bus	roof: protective covering on top of a motor vehicle window: a transparent opening in a vehicle that allows vision out of the sides or back; usually is capable of being opened

Table IV lists the overlaps and the resulting scores for the word pair “car#n#“1 – “bus#n#1“; only gloss permutations with at least one overlap are shown. As for the calculation of the overlap scores, comparing the example gloss of “car#n#1” and the (descriptive) gloss of “bus#n#1” yields a single overlap of the length 1. The score for this overlap is therefore 1^2; i.e. 1. The hyponym gloss of “car#n#1” and the (descriptive) gloss of “bus#n#1” have four overlaps. Three of them have the length 1 while one of the overlaps (“a vehicle”) consists of two words; i.e. it has the length 2. The score is thus calculated as follows: $3*1^2+1*2^2$; i.e. 7. The total similarity score for car#n#1 and bus#n#1 is 615.

TABLE IV. EXAMPLE OF STANDARD LESK OVERLAPS

Tracing Lesk Comparison car#n#1 – bus#n#1			
car#n#1	Bus#n#1	Overlap	Score
Example	Glos	1 x "a"	1
Glos	Glos	1 x "a" 1 x "vehicle"	2
Glos	mero glos	1 x "usually" 1 x "a motor vehicle"	10
hypo glos	Glos	1 x "passengers" 1 x "a vehicle" 1 x "for" 1 x "used"	7
Mero glos	mero glos	1 x "a transparent opening in a vehicle that allows vision out of the sides or back; usually is capable of being opened" 1 x "protective covering on top of a motor vehicle"	505
Total score			615

Generally, two concepts are more closely related the higher the Lesk score is. The Lesk metric, however, is dependent on the number and size of the available glosses. Like other similarity measures, the Lesk score needs reference values to which it can be compared. In other words, the Lesk score needs a threshold value to be meaningful.

Table V compares the Lesk scores for the three word pairs “car#n#1”-“bicycle#n#1,” “car#n#1”-“motorcycle#n#1” and “car#n#1-“bus#n#1”.

TABLE V. UNINTUITIVE RANKING OF CONCEPTS WITH THE LESK SCORE

Pair	Score	Rank
car – bicycle	300	2
car – motorcycle	237	3
car – bus	739	1

Ranking the scored pairs shows that cars and buses are identified as the most similar pair followed by cars and bicycles while cars and motorcycles are identified as the least similar pair. Furthermore, choosing a relevance threshold of 300 implies that cars and bicycles and cars and buses are similar, but cars and motorcycles are not. These results are unintuitive as, following intuition, at least cars and motorcycles should be interpreted as more similar than cars and bicycles since both are motorized vehicles. When regarding shape and function motorcycles and bicycles are similar. This example demonstrates that the Lesk score requires some optimizations to become more meaningful. Optimizations of the Lesk Algorithm are therefore addressed in the following section.

C.3 Optimizations of the Lesk Algorithm

As discussed by Vöhringer & Fliedl et al. in [15], the standard Lesk algorithm has optimization potential in regard to not only internal factors but also in regard to external factors. Optimization of the *internal factors* means that the Lesk algorithm itself is updated and improved while in the optimization of the *external factors* the environment is adapted but the algorithm stays unchanged. The underlying lexicon, in particular, can be optimized regarding contents and structure. Possible optimization strategies include:

- Internal factors:
  - Partial filtering of stop words
  - Word reduction via stemming
  - Normalization based on gloss length
- External factors:
  - Improvement of glossary quality and quantity via completion and substitution of certain keywords
  - Restructuring of taxonomy
  - Guidelines for gloss extension in specific domains

Stop word-based enhancements of the Lesk strategy are, for example, discussed in [32]. A stop word list generated from WordNet can be used for filtering word grams that contain a certain empirically motivated percentage of non-content words. Gloss overlaps above a predefined percentage of stop words are ex-filtrated. Since single stop word overlaps have a 100% stop word quotient they are obligatorily filtered out. For identifying matches of inflected word variants, *stemming* is proposed. In a prototype implementation of the Lesk algorithm, a version of the extended Porter stemmer is used for this purpose [52].

Other Lesk optimization strategies include *normalization by gloss sizes*. Using the standard additive calculation of gloss overlap scores the availability of extensive glosses is naturally favored. If, in contrast, gloss size normalization was used overlap scores would yield the proportional rate of overlaps in a gloss instead of the absolute numbers.

Lesk optimizations with respect to external factors; e.g. by filling gaps in WordNet glosses or restructuring the WordNet taxonomy, are a difficult process. Although there are apparent gaps and suboptimal taxonomies in WordNet that are seemingly easy to improve one must be wary of side effects. The easiest least intrusive change on the external level is the *filling of prominent gloss gaps*; in particular missing hypernyms, hyponyms, meronyms and holonyms. This can be done either manually or by transferring these definitions from other related words. For instance, the two WordNet entries “motorcycle” and “bicycle” miss several fitting meronyms that exist for the related word “car.” Motorcycles, like cars, have an engine, a mirror and a horn. Likewise, “mirror” and “horn” are parts of bicycles too.

As shown in table VI, these already existing glosses are transferred to “motorcycle” and “car” by entering the respective meronyms.

TABLE VI. FILLING GAPS IN WORDNET GLOSSES

Concept	Extension of WordNet standard glosses for meronyms
Car	WordNet standard glosses (not extended)
Bus	WordNet standard glosses (not extended)
Motorcycle	WordNet standard glosses *extended by* • *motorcycle engine*: the engine that propels a motorcycle • *motorcycle horn:* a device on a motorcycle for making a warning noise • *motorcycle mirror*: a mirror that the driver of a motorcycle • *gasoline engine:* an internal-combustion engine that burns gasoline
Bicycle	WordNet standard glosses *extended by* • *bicycle horn:* a device on a bicycle for making a warning noise • *bicycle mirror:* a mirror that the driver of a bicycle can use

To demonstrate the effects of internal and external Lesk optimizations the exemplary word pairs from table V (“car#n#1”-“bicycle#n#1,” “car#n#1”-“motorcycle#n#1” and “car#n#1-“bus#n#1”) have also been tested with updated versions of the Lesk algorithm (see tables VII and VIII). While the original Lesk score listed “car”-“bicycle” as more similar than “car”-“motorcycle” (a score of 300 vs. 237), the internally optimized Lesk implementation yields the scores 115 for the word pair “car”-“bicycle” and 181 for the word pair “car”-“motorcycle.” Thus the improved algorithm using the strategies discussed above better reflects the greater similarity between “car” and “motorcycle.”

TABLE VII. RESULTS OF THE INTERNALLY OPTIMIZED LESK ALGORITHM

Pair	Score	Rank
Car-Bicycle	115	3
Car-Motorcycle	181	2
Car-Bus	688	1

TABLE VIII. RESULTS OF THE INTERNALLY & EXTERNALLY OPTIMIZED LESK ALGORITHM

Pair	Score	Rank
Car-Bicycle	198	3
Car-Motorcycle	321	2
Car-Bus	688	1

Summarized results of the internal and external Lesk optimizations are shown in tables VII and VIII. Obviously, the results in table V don’t adequately reflect the intuitive similarity-ranking concerning form and functionality of the involved entities. Table VII shows that internal optimization already establishes the correct ranking. Table VIII shows the scoring results after filling obvious lexicon gaps with adjusted score distances. The experiment’s results suggest that the outcomes using the optimized Lesk algorithm are more meaningful than the standard Lesk score.

D. Decision on matching results (Step 5)

In short, the following strategies are applied in step 5’s decision on matching results. Both equality and synonymy mean that the compared schema elements match. The integration proposal “matching” is therefore an indication for merging these elements though semantic loss must be avoided under any circumstances. This can be done by storing the original schemata and concept names for traceability reasons. Unrelated schema elements are “dissimilar” and therefore transferred independently to the integrated schema. For (directly) “related” schema elements both elements are transferred to the integrated schema and a relationship between them is introduced. Schema elements are indirectly related when they have no direct connecting relationship in the domain, but are connected via several other concepts. For example, two elements might have a common neighbor concept with which they are connected via generalization- or aggregation-relationships. It is principally possible to also transfer such more complex relationships – including all intermediate concepts – to the integrated schema as a proper connection for the indirectly-related schema elements.

A central requirement says that the integration process should be automatized. This means that domain experts should be supported by preferably accurate proposals and the tool should generate a default-integrated schema even when no user input is made at all. For this purpose an option is provided in the integration tool that allows adjusting the preferred degree of automatization. At its most rigid setting a rough solution is automatically calculated using the matching methods as described in the previous sections if user feedback is absent? The proposals’ quality is influenced by the correctness and completeness of the available domain ontology. If ambiguity conflicts arise they are resolved by automatically choosing the most probable word meaning. While fully automatic integration without any manual input is fast and convenient the quality of the proposed solution is likely to be lower than when user feedback is available. On the other hand, the prototype also allows a setting where the matching and integration is performed stepwise and domain experts need to accept, or reject, the proposals for each schema element pair and each integration step. This setting naturally allows the most direct influence for the user, but it is also the slowest and most laborious. Our recommended strategy is to strike a balance between these two extremes. This can be done by automating the process, but asking the domain experts to provide missing definitions to resolve conflicts like word ambiguities, or contradictions, and to evaluate the end results of the integration.

VI. SUMMARY AND CONCLUSION

In this paper, we have presented a semi-automatic method for matching schema elements in the integration of structural pre-design schemata. Following Rahm & Bernstein et al. [26], our own method can be classified as a composite schema-based matching method. Our approach uses element-level (concept) matching – structural-level (neighborhood) matching and taxonomy-based matching – and combines these parts to one workflow resulting in the proposed integrated schema. The research approach used within this work can be characterized as design science and our main contributions as a method and an instantiation; i.e. a prototype application. When applied in schema integration, our matching method should facilitate the recognition of similarities and differences between two structural source schemata.

REFERENCES

[1] P. Bellström and J. Vöhringer, “A Three-Tier Matching Strategy for Predesign Schema Elements,” Proceedings of The Third International Conference on Information, Process, and Knowledge Management (eKNOW 2011), 2011, pp. 24-29.

[2] A. Doan, F.N. Noy and A.Y. Halevy, “Introduction to the Special Issue on Semantic Integration,” SIGMOD Record, Vol. 33, No. 4, 2004, pp. 11-13.

[3] C. Batini, M.Lenzerini, and S.B. Navathe, “A Compartive Analysis of Methodologies for Database Schema Integration,” ACM Computing Surveys, vol. 18(4), 1986, pp. 323-364.

[4] S. Navathe, R. Elmasri and J. Larson, “Integrating User Views in Database Design,” IEEE Computer 19(1), 1986, 50–62.

[5] Batini, C., S. Ceri and S.B. Navathe, Conceptual Database Design An Entity-Relationship Approach, The Benjamin/Cummings Publishing Company Inc., Redwood City California, 1992.

[6] M. Stumptner, M. Schrefl and G. Grossmann, “On the Road to Behavior-Based Integration,” Proceedings of the 1st APCCM Conference, 2004, pp. 15-22.

[7] A. Bachmann, W. Hesse, A. Russ, C. Kop, H.C., Mayr, and J. Vöhringer J., “OBSE – An Approach to Ontology-Based Software Engineering in the practice,” Proceedings of EMISA, 2007, pp. 129–142.

[8] A.R. Hevner, S.T. March and J. Park, “Design Science in Information Systems Research,” MIS Quarterly, 28 (1), 2004, pp. 75-105.

[9] J. Iivari, “A Paradigmatic Analysis of Information Systems As a Design Science,” Scandinavian Journal of Information Systems, 19 (2), 2007, pp. 39-64.

[10] S.T. March and G.F. Smith “Design and Natural Science Research on Information Technology,” Decision Support Systems, 15, 1995, pp. 251-266.

[11] A. Hevner and S. Chatterjee, Design Research in Information Systems Theory and Practice, New York, Springer (2010).

[12] P. Bellström, View Integration in Conceptual Database Design – Problems, Approaches and Solutions, Licentiate Thesis, Karlstad University Studies 2006:5, 2006.

[13] P. Bellström, Schema Integration – How to Integrate Static and Dynamic Database Schemata, Dissertation, Karlstad University, Karlstad University Studies 2010:12, 2010.

[14] P. Bellström, “A Rule-Based Approach for the Recognition of Similarities and Differences in the Integration of Structural Karlstad Enterprise Modeling Schemata,” Proceedings of the 3rd IFIP WG 8.1 Working Conference on The Practice of Enterprise Modeling (PoEM 2010), 2010, pp. 177-189.

[15] J. Vöhringer and G. Fliedl, “Adapting the Lesk Algorithm for Calculating Term Similarity in the Context of Ontology Engineering,” in Information Systems DevelopmentBusiness Systems and Services: Modeling and Development, 2011, pp. 781-790.

[16] J. Vöhringer, Schema Integration on the Predesign Level, Dissertation, Alpen-Adria-Universität Klagenfurt, 2010.

[17] P. Bellström, J. Vöhringer and C. Kop, “Guidelines for Modeling Language Independent Integration of Dynamic Schemata,” Proceedings of the IASTED International Conference on Software Engineering, 2008, pp. 112-117.

[18] P. Bellström, J. Vöhringer, and C. Kop, “Towards Modeling Language Independent Integration of Dynamic Schemata,” in Information Systems Development Toward a Service Provision Socity, Heidelberg: Springer, 2009, pp. 21-29.

[19] P. Bellström, J. Vöhringer, and A. Salbrechter, “Recognition and Resolution of Linguistic Conflicts: The Core to a Successful View and Schema Integration,” in Advances in Information Systems Development New Methods and Practice for the Networked Society, Vol. 2, 2007, pp. 77-87.

[20] G. Fliedl, C. Kop, H.C. Mayr, W. Mayerthaler and C. Winkler, “Linguistically based requirements engineering – The NIBA project,” Data & Knowledge Engineering, 35, 2000, 111-120.

[21] R. Gustas and P. Gustiené, “Towards the Enterprise Engineering Approach for Information System Modelling Across Organisational and Technical Boundaries,” in Enterprise Information Systems V, Dordrecht: Kluwer, 2004, pp. 204-215.

[22] P. Chen, “The Entity-Relationship Model – Toward a Unified View of Data,” ACM Transactions on Database Systems, vol. 1(1), 1976, pp. 9-36.

[23] W. Song, Schema Integration – Principles, Methods, and Applications, Dissertation, Stockholm University, 1995.

[24] Object Management Group, OMG Unified Modeling Language (OMG UML), Superstructure, [Electronic], Available: http://www.omg.org/spec/UML/2.3/Superstructure/PDF/ [20120126], 2010.

[25] P. Bellström and J. Vöhringer, “Towards the Automation of Modeling Language Independent Schema Integration,” Proceedings of the International Conference on Information, Process, and Knowledge Management (eKNOW 2009), 2009, pp. 110-115.

[26] E. Rahm and P.A. Bernstein, “A Survey of Approaches to Automatic Schema Matching,” The VLDB Journal, vol. 10, 2001, pp. 334–350.

[27] P. Shvaiko and J. Euzenat, “A Survey of Schema-Based Matching Approaches,” Journal of Data Semantics, vol. 4, 2005, pp. 146-171.W.

[28] M.L., Lee and T.W. Ling, “A Methodology for Structural Conflict Resolution in the Integration of Entity-Relationship Schemas,” Knowledge and Information System. 5, 2003, 225-247.

[29] Q. He and T.W. Ling, “Resolvning Schematic Descrepancy in the Integration of Entity-Relationship Schemas,” in: Proceedings of ER 2004, Heidelberg: Springer, pp. 245-258.

[30] Z. Wu and M. Palmer, “Verb semantics and lexical selection,” in Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 1994, pp. 133-138.

[31] G. Hirst and D. St-Onge, “Lexical chains as representations of context for the detection and correction of malapropisms,” in WordNet: An Electronic Lexical Database (Language, Speech, and Communication), 1998.

[32] S. Banerjee and T. Pederson, “An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet,” Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2002), 2002, pp. 136–145.

[33] Wordnet, WordNet A lexical database for English [Electronic], Available: http://wordnet.princeton.edu/ [20120126]

[34] L. Palopoli, G. Terracina, and D. Ursino, “DIKE; A System Supporting the Semi-Automatic Construction of Cooperative Information Systems From Heterogeneous Databases,” Software–Practice and Experiences, vol. 33, 2003, pp. 847-884.

[35] D. Kensche, C. Quix, X. Li, and Y. Li, “GeRoMeSuite: A System for Holistic Generic Model Mangement,” Proceedings of the 33rd International Conference on Very Large Data, 2007, pp. 1322-1325.

[36] H. Frank and K. Eder, “Towards an Automatic Integration of Statecharts,” Proceedings of ER’99, Springer, 1999, pp. 430-445.

[37] P. Shoval, “A Methodology for Integration of Binary-Relationship Conceptual Schemas,” International Conference on Databases, Parallel Architectures and Their Applications, 1990, pp. 435–437.

[38] T.J. Teorey, Database Modeling & Design, Morgan Kaufmann Publishers Inc, USA, 1999.

[39] S. Spaccapietra and C. Parent, “View Integration: a Step Forward in Solving Structural Conflicts,” IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 2, 1994, pp. 258–274.

[40] P. Bellström, “Bridging the Gap between Comparison and Conforming the Views in View Integration,” Local Proceedings of the 10th ADBIS Conference, 2006, pp. 184-199.

[41] P. Johannesson, Schema Integration, Schema Translation, and Interoperability in Federated Information Systems. Dissertation Stockholm University, Royal Institute of Technology, 1993.

[42] L. Ekenberg and P. Johannesson, “A Formal Basis for Dynamic Schema Integration,” Conceptual Modeling – ER’96, Springer, 1996, . pp. 211-226.

[43] P. Bellström, “On the Problem of Semantic Loss in View Integration,” in Information Systems Developent Challenges in Practice, Theory, and Education, Vol. 2, Heidelberg: Springer, 2009, pp. 963-974.

[44] D. Dey, V.C. Story and T.M. Barron, “Improving Database Design through the Analysis of Relationships,” ACM Transactions on Database Systems, 24 (4), 1999, pp. 453-486.

[45] C. Batini and M.Lenzerini, “A Methodology for Data Schema Integration in the Entity-Relationship Model,” IEEE Transactions on Software Engineering, 10 (6), 1984, pp. 650–664.

[46] S.B. Navathe and S.U. Gadgil, “A Methodology for View Integration in Logical Database Design,” Proceedings of the Eighth International Conference on Large Data Bases, Morgan Kaufmann, 1982, pp. 142–164.

[47] Bloomfield, L. (1933). Language. Chicago - London: The University of Chicago Press.

[48] R. Koeling and D. McCarthy, “From Predicting Predominant Sense to Local Context for Word Sense Disambiguation,” Proceedings of the 2008 Conference on Semantics in Text Processing (STEP’08), 2008, pp. 103-114.

[49] K. Heylen, Y. Peirsman, D. Geeraerts and D. Speelman, “Modelling Word Similarity: An Evaluation of Automatic Synonymy Extraction Algorithms,” Proceedings of the 6th International Conference on Language Resource and Evaluation (LREC’08), 2008, pp. 3243-3249.

[50] W. Kim and J. Seo, “Classifying Schematic and Data Heterogeneity in Multidatabase Systems,” IEEE Computer, 24, 1991, pp. 12–18.

[51] J. Ekedahl and K. Golub, Word sense disambiguation using WordNet and the Lesk algorithm. Projektarbeten 2004: Språkbehandling och datalingvistik Lunds Universitet, Institutionen för Datavetenskap, 2005, pp. 17-22.

[52] P. Willet, “The Porter stemming algorithm: then and now,” Electronic Library and Information Systems, 40 (3), 2006, pp. 219-223. ISSN 0033-0337.