On Computing the Importance of Entity Types in Large Conceptual Schemas

Antonio Villegas and Antoni Olivé

Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya
{avillegas,olive}@lsi.upc.edu

Abstract. The visualization and the understanding of large conceptual schemas require the use of specific methods. These methods generate clustered, summarized or focused schemas that are easier to visualize and to understand. All of these methods require computing the importance of each entity type in the schema. In principle, the totality of knowledge defined in the schema could be relevant for the computation of that importance but, up to now, only a small part of that knowledge has been taken into account. In this paper, we extend six existing methods for computing the importance of entity types by taking into account all the relevant knowledge defined in the structural and behavioural parts of the schema. We experimentally evaluate the original and the extended versions of those methods with two large real-world schemas. We present the two main conclusions we have drawn from the experiments.

1 Introduction

Real information systems often have extremely complex conceptual schemas. The visualization and understanding of these schemas require the use of specific methods, which are not needed in small schemas [1]. These methods generate indexed, clustered, summarized or focused schemas that are easier to visualize and to understand [2].

Many of the above methods require computing the importance (also called relevance or score) of each type in the schema. The computed importance induces an ordering of the entity types, which plays a key role in the steps and result (output) of the method. For example, Castano, de Antonellis, Fugini and Pernici [3] propose a three-steps indexing method, in which the first step computes the importance of each entity type, based on the number and kind of relationships it has in the schema. Moody [4] proposes a clustering method in which the most important entity types are hypothesized to be those that have the higher connectivity, defined as the number of relationships in which they participate. Tzitzikas and Hainaut [5,6] propose methods for scoring each entity type in a schema, aiming at facilitating its understanding. As a last example we may mention Yu and Jagadish [7], who propose a metric of the importance of each entity type, which is used in order to automatically produce a good summary of a schema.

Intuitively, it seems that an objective metric of the importance of an entity type in a given schema should be related to the amount of knowledge that the schema defines about it. The more (less) knowledge a schema defines about an entity type, the more (less) important should be that entity type in the schema. Adding more knowledge about an entity type should increase (or at least not decrease) the relative importance of that entity type with respect to the others. Note that in this paper we focus on objective metrics, which are independent from subjective evaluations of users and modelers.

As far as we know, the existing metrics for entity type importance are mainly based on the amount of knowledge defined in the schema, but only take into account the number of attributes, associations and specialization/generalization relationships. Surprisingly, none of the methods we are aware of take into account additional knowledge about entity types defined in a schema that, according to the intuition, could have an effect on the importance. A complete schema [8] includes also cardinalities, taxonomic constraints, general constraints, derivation rules and the specification of events, all of which contribute to the knowledge about entity types.

The main objective of this paper is to analyze the influence of that additional knowledge on a representative set of existing metrics for measuring the importance of the entity types. To this end, we have selected six methods from [3,5,6] and we have developed extended versions of all of them. We have experimentally evaluated both versions of each method using the conceptual schema of the osCommerce [9] and the UML metaschema [10]. The osCommerce is a popular industrial e-commerce system whose conceptual schema consists of 346 entity types (of which 261 are event types). The official 2.0 UML metaschema we have used consists of 293 entity types. The original and the extended versions give exactly the same results from the same input, but the extended versions can process the additional knowledge defined in the schema and then, of course, they give different results. We analyze the differences, and make conclusions on the effect of the additional knowledge on the metrics.

The rest of the paper is organized as follows. Section 2 introduces the concepts and notations. Section 3 briefly describes the seleted methods and explains the extensions we have done to them. Section 4 describes the experimentation with the methods, the results obtained and the conclusions we have drawn. Finally, Section 5 summarizes the paper and points out future work.

2 Basic Concepts and Notations

In this section we review the main concepts and the notation we have used to define the knowlege of conceptual schemas. In this paper, we deal with schemas written in the UML[10]/OCL[11]. Table 1 summarizes the notation (inspired by [6,12]) used in the rest of the paper.

A conceptual schema consists of a structural (sub)schema and a behavioral (sub)schema. The structural schema consists of a taxonomy of entity types (a set of entity types with their generalization/specialization relationships and the

Table 1. Schema Notations

Notation	Definition
$par(e)$	$= \{e' \in \mathcal{E} \mid e \; IsA \; e'\}$
$chi(e)$	$= \{e' \in \mathcal{E} \mid e' \; IsA \; e\}$
$gen(e)$	$= par(e) \cup chi(e)$
$attr(e)$	$= \{a \in \mathcal{A} \mid entity(a) = e\}$
$members(r)$	$= \{e \in \mathcal{E} \mid e$ is a participant of $r\}$
$assoc(e)$	$= \{r \in \mathcal{R} \mid e \in members(r)\}$
$conn(e)$	$= \uplus_{r \in assoc(e)}\{members(r) \backslash \{e\}\}^1$
$context(\alpha)$	$= e \in \mathcal{E} \mid \alpha \in \mathcal{SR} \wedge \alpha \; DefinedIn \; e$
$members(exp)$	$= \{e \in \mathcal{E} \mid e$ is a participant of $exp\}$
$expr(\alpha)$	$= \{expr \mid expr$ is contained in $\alpha\}$
$ref(\alpha)$	$= \cup_{exp \in expr(\alpha)}\{members(exp)\}$
$expr_{nav}(\alpha)$	$= \{expr \in expr(\alpha) \mid expr$ is a navigation expression$\}$
$nav_{expr}(\alpha)$	$= \cup_{exp \in expr_{nav}(\alpha)}\{\{e, e'\} \subset \mathcal{E} \mid \{e, e'\} = members(exp)\})$
$nav_{context}(\alpha)$	$= \{\{e, e'\} \subset \mathcal{E} \mid e = context(\alpha) \wedge e' \in ref(\alpha)\}$
$nav(\alpha)$	$= nav_{context}(\alpha) \cup \; nav_{expr}(\alpha)$
$rconn(e)$	$= \uplus_{\alpha \in \mathcal{SR}}\{e' \in \mathcal{E} \mid \{e, e'\} \subset nav(\alpha)\}$
$par_{inh}(e)$	$= par(e) \cup \{par_{inh}(e') \mid e' \in par(e)\}$
$chi_{inh}(e)$	$= chi(e) \cup \{chi_{inh}(e') \mid e' \in chi(e)\}$
$attr_{inh}(e)$	$= attr(e) \cup \{attr_{inh}(e') \mid e' \in par(e)\}$
$assoc_{inh}(e)$	$= assoc(e) \uplus \{assoc(e') \mid e' \in par_{inh}(e)\}$
$conn_{inh}(e)$	$= conn(e) \uplus \{conn(e') \mid e' \in par_{inh}(e)\}$
$rconn_{inh}(e)$	$= rconn(e) \uplus \{rconn(e') \mid e' \in par_{inh}(e)\}$

taxonomic constraints), a set of relationship types (either attributes or associations), the cardinality constraints of the relationship types, and a set of other static constraints formally defined in OCL.

We denote by $\mathcal{E}$ the set of entity types defined in the schema. For a given $e \in \mathcal{E}$ we denote by $par(e)$ and $chi(e)$ the set of directly connected ascendants and descendants of e, respectively, and by $gen(e)$ the union of both sets. The set of attributes defined in the schema is denoted by $\mathcal{A}$. If $a \in \mathcal{A}$ then $entity(a)$ denotes the entity type where a is defined. The set of attributes of an entity type e is denoted by $attr(e)$.

The set of associations defined in the schema is denoted by $\mathcal{R}$. If $r \in \mathcal{R}$ then $members(r)$ denotes the set of entity types that participate in association r, and $assoc(e)$ the set of associations in which e participates. Note that an entity type e may participate more than once in the same association, and therefore $members(r)$ and $assoc(e)$ are multisets (may contain duplicate elements). Moreover, $conn(e)$ denotes the multiset of entity types connected to e through associations. For example, if r_1 is the association HasComponent(assembly:Part, component:Part), then

[1] Note that "\" denotes the difference operation of multisets as in $\{a, a, b\} \backslash \{a\} = \{a, b\}$ and "$\uplus$" denotes the multiset (or bag) union that produces a multiset as in $\{a, b\} \uplus \{a\} = \{a, a, b\}$.

$members(r_1)$={Part, Part}, $assoc$(Part)={HasComponent, HasComponent} and $conn$(Part)={Part}.

The behavioural schema consists of a set of event types. We adopt the view that events can be modeled as a special kind of entity type. Event types have characteristics, constraints and effects. The characteristics of an event are its attributes and the associations in which it participates. The constraints are the conditions that events must satisfy to occur. Each event type has an operation called *effect()* that gives the effect of an event occurence. The effect is declaratively defined by the postcondition of the operation, which is specified in OCL (see chp. 11 of [8]). Furthermore, entity and relationship types may be base or derived. If they are derived, there is a formal derivation rule in OCL that defines their population in terms of the population of other types.

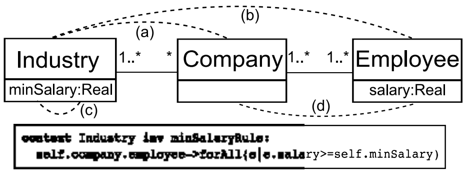
We denote by $\mathcal{SR}$ the set of constraints, derivation rules and pre- and postconditions. Each schema rule α is defined in the context of an entity type, denoted by $context(\alpha)$. In OCL, each rule α consists of a set of OCL expressions (see OCL [11]) which we denote by $expr(\alpha)$. An expression exp may refer to several entity types which are denoted by $members(exp)$. The set of entity types that are referred to in one or more expressions of a rule α is denoted by $ref(\alpha)$.

We also include in $\mathcal{SR}$ the schema rules corresponding to the equivalent OCL invariants of the cardinality constraints. For example, in Fig. 1 the cardinality "1.." between Company and Employee is transformed into the invariant:

```
context Company inv: self.employee->size()>0
```

A special kind of OCL expression is the navigation expression that define a schema navigation from an entity type to another through an association (see *NavigationCallExp* of OCL in [11]). We use $expr_{nav}(\alpha)$ to indicate the navigation expressions inside a rule α. Such expressions only contain two entity types as its participants, i.e. the *source* entity type and the *target* one (see the example in Fig. 1).

We denote by $nav_{expr}(\alpha)$ the set of pairs that participate in the navigation expressions of α. We also denote by $nav_{context}(\alpha)$ the sets of pairs of entity types composed by the context of the rule α and every one of the participant entity types of such rule ($e \in ref(\alpha)$). Finally, we define $nav(\alpha)$ as the union of $nav_{context}(\alpha)$ with $nav_{expr}(\alpha)$ and, $rconn(e)$ as the multiset of entity types



```
context Industry inv minSalaryRule:
  self.company.employee->forAll(e|e.salary>=self.minSalary)
```

$$
\begin{aligned}
context(\text{minSalaryRule}) &= \text{Industry}\\
expr_{nav}(\text{minSalaryRule}) &= \{\text{self.company}, \text{company.employee}\}\\
ref(\text{minSalaryRule}) &= \{\text{Industry, Company, Employee}\}\\
nav_{context}(\text{minSalaryRule}) &= \{\{\text{Industry, Industry}\}, \{\text{Industry, Company}\}, \{\text{Industry, Employee}\}\}\\
nav_{expr}(\text{minSalaryRule}) &= \{\{\text{Industry, Company}\}, \{\text{Company, Employee}\}\}\\
nav(\text{minSalaryRule}) &= \{\{\text{Industry, Industry}\}, \{\text{Industry, Company}\}, \{\text{Company, Employee}\}, \{\text{Industry, Employee}\}\}
\end{aligned}
$$

Fig. 1. Example of navigations of minSalaryRule. Dashed lines (a), (b) and (c) represent the elements in $nav_{context}$(minSalaryRule) while (d) and (a) are the connections through navigation expressions (see nav_{expr}(minSalaryRule)).

that compose a pair with e in $nav(\alpha)$. Note that since we use $\uplus$, $rconn(e)$ may contain duplicates because it takes into account each rule α and an entity type e can be related to another one e' in two or more different rules. Intuitively, $rconn(e)$ is the multiset of entity types to which an entity type e is connected through schema rules.

The last row section in Table 1 defines the notation we use to take into account the inherited properties from the ancestors of entity types. As a special case, $chi_{inh}(e)$ is the set of descendants of e.

3 Methods and Their Extensions

In this section we briefly review the definition of six existing methods for computing the importance of entity types in a schema. Each method is followed by a brief description and formal definition of our extension to it.

The original version of the methods only takes into account the indicated elements of the structural schema while in the extended version we also take into account the rules and the complete behavioural schema.

3.1 The Simple Method

This method was introduced in [6] and takes into account only the number of directly connected elements. Formally, the importance $I_{SM}(e)$ of an entity type e is defined as:

$$I_{SM}(e) = |par(e)| + |chi(e)| + |attr(e)| + |assoc(e)|$$

Our extension to this method follows the same idea but also including the number of participations of each entity type in the navigation relationships represented in the schema rules specification, i. e., derivation rules, invariants and pre- and postconditions (and cardinality constraints). On the other hand, we now take into account (in $|assoc(e)|$) the associations of each entity type e with the event types of the behavioural schema. Formally:

$$I^{+}_{SM}(e) = |par(e)| + |chi(e)| + |attr(e)| + |assoc(e)| + |rconn(e)|$$

For example, in the schema shown in Fig.1 we would have I_{SM}(Company)=2 and I^{+}_{SM}(Company)=8, because $|par$(Company)$|$=$|chi$(Company)$|$=$|attr$(Company)$|$ =0, $|assoc$ (Company)$|$=2, and $|rconn$(Company)$|$=6, of which two come for the invariant (minSalaryRule) and the other four from the OCL equivalent to the cardinality constraints of multiplicity "1..*" in its relationships with Industry and Employee.

3.2 The Weighted Simple Method

This is a variation to the simple method that assigns a strength to each kind of component of knowledge in the equation, such that the higher the strength, the greater the importance of such component [3]. The definition of importance here is:

$$I_{WSM}(e) = q_{inh}(|par(e)| + |chi(e)|) + q_{attr}|attr(e)| + q_{assoc}|assoc(e)|$$

where q_{attr} is the strength for attributes, q_{inh} is the strength for generalization/specialization relationships, and q_{assoc} is the strength for associations. Each of them with values in the interval *[0,1]*.

Our extension to this method consists on adding the *schema rules navigation* component to the importance computation. In the same way as the other components, we selected a strength (q_{rule}) to specify the weight of navigation relationships in the schema rules. The definition is now:

$$I^{+}_{WSM}(e) = q_{inh}(|par(e)| + |chi(e)|) + q_{attr}|attr(e)| + q_{assoc}|assoc(e)| + q_{rule}|rconn(e)|$$

3.3 The Transitive Inheritance Method

This is a variation of the simple method taking into account both directly defined features and inherited ones [6]. For each entity type the method computes the number of ascendants and descendants and all specified attributes and accessible associations from it or any of its ascendants. Formally:

$$I_{TIM}(e) = |par_{inh}(e)| + |chi_{inh}(e)| + |attr_{inh}(e)| + |assoc_{inh}(e)|$$

In the same way as before, we extend it with the *schema rules navigation* component. This time the computation of such component also takes into account the *rconn* of the ancestors:

$$I^{+}_{TIM}(e) = |par_{inh}(e)| + |chi_{inh}(e)| + |attr_{inh}(e)| + |assoc_{inh}(e)| + |rconn_{inh}(e)|$$

3.4 EntityRank

The EntityRank method [5,6] is based on link analysis following the same approach than Google's PageRank [13]. Roughly, each entity type is viewed as a state and each association between entity types as a bidirectional transition between them.

The importance of an entity type is the probability that a random surfer is at that entity type with random jumps (q component) or by navigation through relationships ($1-q$ component). Therefore, the resulting importance of the entity types correspond to the stationary probabilities of the Markov chain, given by:

$$I_{ER}(e) = \frac{q}{|\mathcal{E}|} + (1-q) \sum_{e' \in conn(e)} \frac{I_{ER}(e')}{|conn(e')|}$$

In our extension to it we add a new component to the formula in order to jump not only to the connected entity types but also to the virtually connected ones through the navigation relationships uncovered in the schema rules. The definition is now:

$$I^+_{ER}(e) = \frac{q}{|\mathcal{E}|} + (1-q)\left(\sum_{e' \in conn(e)} \frac{I^+_{ER}(e')}{|conn(e')|} + \sum_{e'' \in rconn(e)} \frac{I^+_{ER}(e'')}{|rconn(e'')|}\right)$$

3.5 BEntityRank

This is a variation of the previous method specifying that the probability of randomly jumping to each entity type is not the same for each entity type, but it depends on the number of its attributes [5,6]. The higher the number of attributes, the higher the probability to randomly jump to that entity type. That is:

$$I_{BER}(e) = q\frac{attr(e)}{|\mathcal{A}|} + (1-q)\sum_{e' \in conn(e)} \frac{I_{BER}(e')}{|conn(e')|}$$

Our extension is in the same way as in EntityRank but taking into account the definition of the attributes component of BEntityRank. The definition is:

$$I^+_{BER}(e) = q\frac{attr(e)}{|\mathcal{A}|} + (1-q)\left(\sum_{e' \in conn(e)} \frac{I^+_{BER}(e')}{|conn(e')|} + \sum_{e'' \in rconn(e)} \frac{I^+_{BER}(e'')}{|rconn(e'')|}\right)$$

3.6 CEntityRank

Finally, the method that we call CEntityRank (m_4 in [6]) follows the same idea than EntityRank and BEntityRank, but including the generalization relationships. Each generalization between ascendants and descendants is viewed as a bidirectional transition, as in the case of associations. Formally:

$$I_{CER}(e) = q_1\frac{attr(e)}{|\mathcal{A}|} + q_2\sum_{e' \in gen(e)} \frac{I_{CER}(e')}{|gen(e')|} + (1-q_1-q_2)\sum_{e'' \in conn(e)} \frac{I_{CER}(e'')}{|conn(e'')|}$$

One more time, our extension includes the uncovered navigations of the schema rules as bidirectional transitions for the random surfer. The new definition is:

$$\begin{aligned} I^+_{CER}(e) = {} & q_1\frac{attr(e)}{|\mathcal{A}|} + q_2\sum_{e' \in gen(e)} \frac{I^+_{CER}(e')}{|gen(e')|} \\ & + (1-q_1-q_2)\left(\sum_{e'' \in conn(e)} \frac{I^+_{CER}(e'')}{|conn(e'')|} + \sum_{e''' \in rconn(e)} \frac{I^+_{CER}(e''')}{|rconn(e''')|}\right) \end{aligned}$$

4 Experimental Evaluation

We have implemented the six methods described in the previous section, in both the original and the extended versions. We have then evaluated the methods using two distinct case studies: the osCommerce [9] and the UML metaschema. The original methods have been evaluated with the input knowledge they are able to process: the entity types, attributes, associations and generalization/specialization relationships of the structural schemas.

For the osCommerce, the extended versions have been evaluated with the complete structural schema, and the complete behavioural schema (including event types and their pre/post conditions). The osCommerce schema comprises 346 entity types (of which 261 are event types), 458 attributes, 183 associations, 204 general constraints and derivation rules and 220 pre- and post conditions. For the UML metaschema there is no behavioral schema and therefore we have only used the complete structural schema. The version of the UML metaschema we have used comprises 293 entity types, 93 attributes, 377 associations, 54 derivation rules and 116 general constraints. In the following, we summarize the two main conclusions we have drawn from the study of the result data.

4.1 Correlation between the Original and the Extended Versions

Figure 2 shows, for each method, the results obtained in the original and the extended versions for the osCommerce. The horizontal axis has a point for each of the 85 entity types of the structural schema, ordered descendently by their importance in the original version. The vertical axis shows the importance computed in both versions. The importance has been normalized such that the sum of the importances of all entity types in each method is 100.

As shown in Fig. 2(f) the highest correlation between the results of both versions is for the CEntityRank (r=0.931), closely followed by the BEntityRank (r=0.929). The lowest correlation is for the Weighted Simple Method (r=0.61).

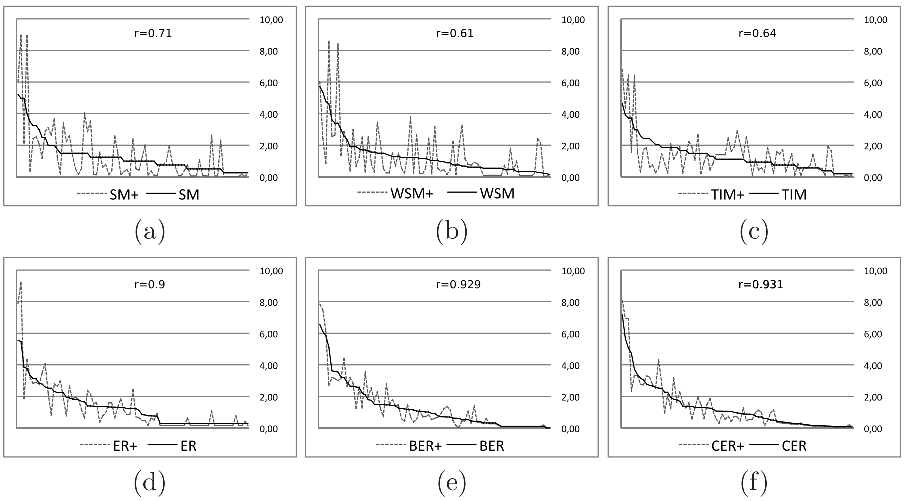


Fig. 2. Comparison between base and extended importance-computing methods once applied to the osCommerce schema

Similar results are obtained for the UML metamodel. In this case the correlation between the two versions of the Weighted Simple Method is 0.84 and that of the CEntityRank is 0.95.

The conclusion from this result is that the method that produces more similar results in both versions is the CEntityRank, followed by the BEntityRank. The conclusion is significant because it implies that if we have to compute the importance of the entity types of a schema, but we only have its attributes, associations and generalization/specialization relationships, the original method that gives results more similar to those that would be obtained in the extended method is the CEntityRank, followed by the BEntityRank. We tend to believe that these are the methods of choice when one wants to compute the relative importance of entity types taking into account the whole schema, but only a fragment of it is available (or only a fragment of it can be processed with the available tools).

This conclusion contrasts with the results reported in [6], which, based on subjective evaluations given by evaluators, concludes that the method that gives the best results is the Simple Method. However, Fig. 2(a) shows that the result given by that method considerably changes when the whole schema knowledge is taken into account.

4.2 Variability of the Original and the Extended Versions

Table 2 shows the correlation between each pair of methods (separately, originals and extended versions), in both case studies. It can be seen that, if we exclude the Transitive Inheritance Method (TIM) because it gives the worst results, the correlation in the original versions of the methods ranges from 0.59 to 0.98, while in the extended versions the range is from 0.83 to 0.99.

The conclusion from this result is that the extended versions of the methods, excluding TIM, produce remarkably similar results, which does not happen in the original version. This conclusion is also significant because it assures that

Table 2. Correlation coefficients between results of original and extended methods

UML Metaschema

	I_{WSM}	I_{TIM}	I_{ER}	I_{BER}	I_{CER}
I_{SM}	0.98	0.15	0.82	0.79	0.92
I_{WSM}		0.16	0.73	0.77	0.90
I_{TIM}			0.06	0.07	0.11
I_{ER}				0.82	0.83
I_{BER}					0.91

	I^+_{WSM}	I^+_{TIM}	I^+_{ER}	I^+_{BER}	I^+_{CER}
I^+_{SM}	0.99	0.26	0.93	0.83	0.86
I^+_{WSM}		0.27	0.91	0.85	0.89
I^+_{TIM}			0.25	0.24	0.30
I^+_{ER}				0.84	0.84
I^+_{BER}					0.91

osCommerce

	I_{WSM}	I_{TIM}	I_{ER}	I_{BER}	I_{CER}
I_{SM}	0.97	0.79	0.74	0.87	0.88
I_{WSM}		0.79	0.59	0.78	0.76
I_{TIM}			0.40	0.54	0.61
I_{ER}				0.94	0.94
I_{BER}					0.97

	I^+_{WSM}	I^+_{TIM}	I^+_{ER}	I^+_{BER}	I^+_{CER}
I^+_{SM}	0.99	0.79	0.98	0.93	0.93
I^+_{WSM}		0.79	0.98	0.94	0.94
I^+_{TIM}			0.78	0.73	0.83
I^+_{ER}				0.94	0.93
I^+_{BER}					0.97

the use of the Simple Method (extended version) whose computational cost is very low, and on the other hand it allows the incremental recalculation of the importance of entity types when the schema changes, produces "good-enough" results.

5 Conclusions and Further Work

The visualization and the understanding of large conceptual schemas require the use of specific methods. These methods generate indexed, clustered, summarized or focused schemas that are easier to visualize and understand. Almost all of these methods require computing the importance of each entity type in the schema. We have argued that the objective importance of an entity type in a schema should be related to the amount of knowledge that the schema defines about it. There are several proposals of metrics for entity type importance. All of them are mainly based on the amount of knowledge defined in the schema, but -surprisingly- they only take into account the fragment of that knowledge consisting on the number of attributes, associations and specialization/generalization relationships. A complete conceptual schema also includes cardinalities, general constraints, derivation rules and the specification of events, all of which contribute to the knowledge of entity types.

We have analyzed the influence of that additional knowledge on a representative set of six existing metrics. We have developed extended versions of each of those metrics. We have evaluated both versions of those methods in two large real-world schemas. The two main conclusions are: (1) Among the original versions of the methods, the methods of choice are those based on the link analysis following the same approach than Google's PageRank; and (2) The extended versions of most methods produce remarkably similar results, which does not happen in the original version.

We plan to continue this work in two main directions. The first, is the experimentation with other large industrial schemas to check whether the above conclusions may have a larger experimental basis. The second, is the extension of the work to other existing metrics.

Acknowledgements. Thanks to the anonymous referees and the people of the GMC group for their useful comments to previous drafts of this paper. This work has been partly supported by the Ministerio de Ciencia y Tecnologia under TIN2008-00444 project, Grupo Consolidado.

References

1. Olivé, A., Cabot, J.: A research agenda for conceptual schema-centric development. In: Krogstie, J., Opdahl, A.L., Brinkkemper, S. (eds.) Conceptual Modelling in Information Systems Engineering, pp. 319–334. Springer, Heidelberg (2007)
2. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding quality in conceptual modeling. IEEE Software 11(2), 42–49 (1994)

3. Castano, S., Antonellis, V.D., Fugini, M.G., Pernici, B.: Conceptual schema analysis: Techniques and applications. ACM Trans. Database Syst. 23(3), 286–332 (1998)
4. Moody, D.L., Flitman, A.: A Methodology for Clustering Entity Relationship Models – A Human Information Processing Approach. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 114–130. Springer, Heidelberg (1999)
5. Tzitzikas, Y., Hainaut, J.L.: How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms). In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 144–159. Springer, Heidelberg (2005)
6. Tzitzikas, Y., Kotzinos, D., Theoharis, Y.: On ranking rdf schema elements (and its application in visualization). J. UCS 13(12), 1854–1880 (2007)
7. Yu, C., Jagadish, H.V.: Schema summarization. In: Dayal, U., Whang, K.Y., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y.K. (eds.) VLDB, pp. 319–330. ACM, New York (2006)
8. Olivé, A.: Conceptual Modeling of Information Systems. Springer, Heidelberg (2007)
9. Tort, A., Olivé, A.: The osCommerce Conceptual Schema. Universitat Politècnica de Catalunya (2007), `http://guifre.lsi.upc.edu/OSCommerce.pdf`
10. Object Management Group (OMG): Unified Modeling Language (UML) Superstructure Specification, version 2.2 (February 2009)
11. Object Management Group (OMG): Object Constraint Language Specification (OCL), version 2.0 (May 2006)
12. Baroni, A.L.: Formal definition of object-oriented design metrics. Master's thesis, Vrije Universiteit Brussel (2002)
13. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: Computer Networks and ISDN Systems, pp. 107–117. Elsevier Science Publishers B. V., Amsterdam (1998)