

DEPARTAMENT DE LENGUATGES I SISTEMES INFORMÀTICS
MASTER IN COMPUTING

MASTER THESIS

COMPUTING THE IMPORTANCE
OF SCHEMA ELEMENTS
TAKING INTO ACCOUNT
THE WHOLE SCHEMA

Student: Antonio Villegas Niño
Director: Antoni Olivé Ramon



JUNE 2009
UNIVERSITAT POLITÈCNICA DE CATALUNYA

<i>Title</i>	Computing the Importance of Schema Elements Taking into Account the Whole Schema
<i>Author</i>	Antonio Villegas Niño
<i>Advisor/Director</i>	Antoni Olivé Ramon
<i>Date</i>	June 2009
<i>Abstract</i>	<p>Conceptual Schemas are one of the most important artifacts in the development cycle of information systems. To understand the conceptual schema is essential to get involved in the information system that is described within it. As the information system increases its size and complexity, the relative conceptual schema will grow in the same proportion making difficult to understand the main concepts of that schema/information system.</p> <p>The thesis comprises the investigation of the influence of the whole schema in computing the relevance of schema elements. It will include research and implementation of algorithms for scoring elements in the literature, an study of the different results obtained once applied to a few example conceptual schemas, an extension of those algorithms including new components in the computation process like derivation rules, constraints and the behavioural subschema specification, and an in-depth comparison among the initial algorithms and the extended ones studying the results in order to choose those algorithms that give the most valuable output.</p>
<i>Keywords</i>	Large Conceptual Schemas, Relevance Computing, UML, OCL
<i>Language</i>	English
<i>Modality</i>	Research Work

*To those who believe in me
and specially to my family
my girlfriend, and closer friends
for their patience*

Acknowledgments

Thanks to my master thesis director Dr. Antoni Olivé for his support and for give me the opportunity of discover the great world of research.

Thanks to my colleagues in the *Grup de Recerca en Modelització Conceptual* (GMC) for let me participate in their activities and interesting meetings.

Thanks to Miquel Camprodon for the interesting conversations about maths and statistics and thanks to David Aguilera for all these years working together.

An also thanks to the L^AT_EXcommunity for the big amounts of documentation that made easy to write this document.

This work has been partly supported by the Ministerio de Ciencia y Tecnología under TIN2008-00444 project, Grupo Consolidado.

This work has been partly supported by the Universitat Politècnica de Catalunya under Becas UPC Recerca program.

Scientific contributions

The results of this work have been submitted and accepted for publication in the proceedings of the ER 2009 affiliated workshop CoMoL (Conceptual Modeling in the Large):

Antonio Villegas, Antoni Olivé. On Computing the Importance of Entity Types in Large Conceptual Schemas. In *Proceedings of Conceptual Modeling in the Large (CoMoL 2009). 28th International Conference on Conceptual Modeling (ER 2009) affiliated workshop. Gramado, Brazil*. Lecture Notes in Computer Science, Springer-Verlag

Contents

1	Introduction	1
1.1	Conceptual Modeling	1
1.2	Large Conceptual Schemas	2
1.3	Aim of this Thesis	3
1.4	Outline of the document	4
2	Background	5
2.1	Conceptual Schemas	5
2.1.1	Structural Schema	7
2.1.2	Behavioural Schema	8
2.2	Modeling Languages	10
2.2.1	The Entity-Relationship Model	10
2.2.2	The Unified Modeling Language	11
2.2.3	The Object Constraint Language	12
2.3	Information Extraction	14
2.4	Human Capacity for Processing Information	15
3	State of the Art	17
3.1	Requests for Contributions	17
3.2	Clustering	18
3.3	Filtering/Scoring/Ranking	29
3.4	Metrics	35
3.5	Visualization	36
3.6	Conclusions	41
4	Measures, Methods and Extensions	43
4.1	Measures	43
4.1.1	Importance-computing Principles	44
4.1.2	Basic Metrics	45
4.2	Methods	47
4.2.1	The Connectivity Counter (CC)	48
4.2.2	The Simple Method (SM)	49
4.2.3	The Weighted Simple Method (WSM)	49
4.2.4	The Transitive inheritance Method (TIM)	50
4.2.5	EntityRank (ER)	50
4.2.6	BEntityRank (BER)	51
4.2.7	CEntityRank (CER)	52
4.3	Extended Measures	53

4.3.1	Complex Relationship Types	54
4.3.2	The Power of OCL	57
4.3.3	Extended metrics	61
4.4	Extended Methods	63
4.4.1	The Connectivity Counter Extended (CC+)	64
4.4.2	The Simple Method Extended (SM+)	64
4.4.3	The Weighted Simple Method Extended (WSM+)	65
4.4.4	The Transitive inheritance Method Extended (TIM+)	66
4.4.5	EntityRank Extended (ER+)	66
4.4.6	BEntityRank Extended (BER+)	67
4.4.7	CEntityRank Extended (CER+)	68
4.5	Comparison	69
5	Experimental Evaluation	73
5.1	Conceptual Schemas for the Evaluation	73
5.1.1	The osCommerce	74
5.1.2	The UML Metaschema	77
5.2	Correlation Study	78
5.2.1	Correlation Between the Original and the Extended Versions	78
5.2.2	Variability of the Original and the Extended Versions	82
5.3	Timing Evaluation	88
5.4	Evaluation of Results	91
6	Implementation	95
6.1	The USE Tool	96
6.2	The Architecture	97
6.3	Computing Link Analysis based Methods	99
6.3.1	The Power of Linear Algebra	99
6.3.2	EntityRank (ER)	101
6.3.3	BEntityRank (BER)	102
6.3.4	CEntityRank (CER)	103
6.3.5	EntityRank Extended (ER+)	104
6.3.6	BEntityRank Extended (BER+)	104
6.3.7	CEntityRank Extended (CER+)	104
6.4	User Manual	105
6.4.1	Working with USE	105
6.4.2	Compute the Relevance of Entity Types	107
6.4.3	Visualization of Results	109
7	Conclusions	111
7.1	Conceptual Modeling	111
7.2	State of the Art	112
7.3	Thesis Contribution	113
7.4	Future Work	114
	Bibliography	117

List of Figures

2.1	Classification of domain concepts into entity types.	6
2.2	Example of event types.	9
2.3	ER diagram about customers.	10
2.4	Entity-Relationship diagram.	11
2.5	Class diagram of the 13 types of diagrams of the UML 2.0. . . .	12
2.6	Example of conceptual schema specified in UML 2.0	13
2.7	Example of schema rules specified in OCL 2.0	13
2.8	Tag cloud of the superstructure specification of the UML 2.0 [39].	15
2.9	Bottleneck in human and computer capacity for processing information and solutions. While computers can be replicated, each human being only can have a brain, so the solution is to reduce the amount of information to process.	16
3.1	Three levels of abstraction diagramming. Extracted from [20]. . .	19
3.2	Example of dominance grouping operation. Extracted from [51].	21
3.3	Example of Entity Cluster Levels. Extracted from [51].	22
3.4	Example of Relationship Clustering. Extracted from [28].	23
3.5	Example of Level Entity Relationship Model. Extracted from [22].	24
3.6	Example of Levelled Data Model. Extracted from [37, 36].	26
3.7	Manual Decomposition Procedure. Extracted from [37, 36]. . . .	28
3.8	Application of clustering algorithm of Tavana, Joglekar and Redmond. Extracted from [50].	29
3.9	Representative elements of a conceptual schema. Extracted from [13].	31
3.10	PageRank example.	32
3.11	EntityRank application. Extracted from [54].	33
3.12	3D mesh representing a plane at four different levels of detail. Extracted from [49].	37
3.13	Blur effect in Focus+Context technique to highlight persons. . .	38
3.14	Example of visualization techniques.	38
3.15	Example of Indented List and Node-Link Tree.	39
3.16	Example of World's Population visualized with a Tree Map, included in Space Filling techniques.	40
3.17	Example of 3D landscape of a Unix File System.	40
4.1	Example of basic metrics.	47
4.2	Example of schema.	48
4.3	Implicit reification of a binary relationship with association class.	54

4.4	Implicit reification of a n -ary ($n > 2$) relationship with association class.	55
4.5	Implicit reification of a ternary relationship with association class.	56
4.6	Fragment of the OCL metamodel including and overview of the expressions. Extracted from [38].	58
4.7	Fragment of the OCL metamodel including navigation expressions. Extracted from [38].	58
4.8	Example of uncovered links extracted from the OCL.	59
4.9	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 $nav_{expr}(minSalaryRule)$).	62
4.10	Extension of the schema in Fig.4.2 with some OCL invariants.	63
5.1	Order confirmation at osCommerce.	74
5.2	Simplification of the structural schema of the osCommerce. Extracted from [52]	75
5.3	UpdateOrderStatus event at osCommerce. Extracted from [52]	76
5.4	Shape of the Kernel section of the UML metaschema.	77
5.5	Comparison between base and extended importance-computing methods once applied to the osCommerce schema.	80
5.6	Comparison between base and extended importance-computing methods once applied to the UML metaschema.	81
5.7	Comparison between base methods applied to the UML metaschema.	84
5.8	Comparison between extended methods applied to the UML metaschema.	85
5.9	Comparison between base methods applied to the osCommerce.	86
5.10	Comparison between extended methods applied to the osCommerce.	87
5.11	Values in milliseconds (ms) of execution time.	89
5.12	Values in milliseconds (ms) of execution time per entity type.	90
6.1	Snapshot of the USE tool.	96
6.2	Architecture design for the implementation of the relevance computing methods.	97
6.3	Steps to compute the importance of the entity types belonging to a conceptual schema.	98
6.4	Example of simple conceptual schema.	99
6.5	USE main window.	105
6.6	Open a conceptual schema in USE.	106
6.7	Class diagram view of the osCommerce schema in USE.	107
6.8	Selection of the importance computing method in USE.	108
6.9	Saving the results of an importance computing method in USE.	108
6.10	Ranking list of the osCommerce in USE.	109
6.11	Top-20 entity types of the osCommerce in USE.	110
6.12	Top-10 entity types of the osCommerce in USE.	110

List of Tables

3.1	Some metrics introduced by Baroni et al. in [4, 5, 6, 7].	35
3.2	Metrics introduced by Genero et al. in [25].	36
4.1	Definition of basic metrics.	46
4.2	Results for CC applied to example of Fig 4.2.	48
4.3	Results for SM applied to example of Fig 4.2.	49
4.4	Results for WSM applied to example of Fig 4.2.	50
4.5	Results for TIM applied to example of Fig 4.2.	50
4.6	Results for ER applied to example of Fig 4.2.	51
4.7	Results for BER applied to example of Fig 4.2.	52
4.8	Results for CER applied to example of Fig 4.2.	53
4.9	Definition of extended metrics.	62
4.10	Results for CC+ applied to example of Fig 4.10.	64
4.11	Results for SM+ applied to example of Fig 4.10.	65
4.12	Results for WSM+ applied to example of Fig 4.10.	66
4.13	Results for TIM+ applied to example of Fig 4.10.	67
4.14	Results for ER+ applied to example of Fig 4.10.	67
4.15	Results for BER+ applied to example of Fig 4.10.	68
4.16	Results for CER+ applied to example of Fig 4.10.	68
4.17	Classification of selected methods according to their approach.	69
4.18	Comparison of knowledge used between both base and extended versions of the selected methods.	71
5.1	Connectivity degree δ for the test schemas.	78
5.2	Correlation coefficients between results of original and extended methods for the UML metaschema.	82
5.3	Correlation coefficients between results of original and extended methods for the osCommerce.	83
5.4	Values in milliseconds (<i>ms</i>) of execution time.	88
5.5	Values in milliseconds (<i>ms</i>) of execution time per entity type.	90
5.6	Comparison between the top-10 of the methods for the osCommerce.	92
5.7	Comparison between the top-10 of the methods for the UML metaschema.	93
5.8	Comparison between the top-10 event types of the methods for the osCommerce.	94
6.1	Eigenvalues and associated eigenvectors	102

6.2 Ranking of entity types for the example of Fig. 6.4 using the
linear algebra version of the ER. 102

COMPUTING THE IMPORTANCE
OF SCHEMA ELEMENTS
TAKING INTO ACCOUNT
THE WHOLE SCHEMA

Chapter 1

Introduction

This chapter starts with a short overview of conceptual modeling of information systems and its role in the software engineering process in Section 1.1. Special attention is given to the problem of dealing with large conceptual schemas which is presented in Section 1.2.

The research interest of this thesis is related with methods to compute the importance of schema elements and their application in the visualization and usability improvement of conceptual schemas. A brief description of the objectives is presented in Section 1.3 and, finally, Section 1.4 finishes the chapter with an outline of the rest of the thesis.

1.1 Conceptual Modeling

Conceptual modeling can be roughly defined as the activity that must be done to obtain the conceptual schema of an information system. We define information system as a designed system that collects, stores, processes and distributes information about the state of a domain.

Thus, conceptual modeling is one of the initial activities in the software engineering process and consists of gather, classify, structure and maintain knowledge about a real-world domain. This processed knowledge conforms the conceptual schema.

Conceptual schemas are the central unit of knowledge in the development process of information systems. A conceptual schema must include the definition of all relevant characteristics appearing in an organization that are useful in the task of representation (also know as conceptualization) of the information system.

Usually, conceptual schemas have been seen as documentation items in the software engineering process. However, since the initial stages of model-driven

approaches [31] came out many years ago, conceptual schemas have increased its importance and participation as key artifacts in the software development activities. Model-driven approaches try to generate software (including both code and documentation) from a specified model, which may be a conceptual schema of an organization domain. This new role given to conceptual schemas implies that the specification and comprehension of conceptual schemas are main tasks for the stakeholders of an information system.

A conceptual schema provides an abstraction layer between the real-world knowledge and the portion of that knowledge that is really useful in the development of an information system. This abstraction provides simplification to describe real concepts as general ones without taking into account the development technology of the final stages in the software engineering process.

The development process of information systems always include a conceptual schema. Sometimes, such schema can be explicitly reproduced as a piece of documentation and, sometimes, the schema is shared in the minds of the stakeholders. In any case, the conceptual schema of an information system exists, although obviously to have the schema explicitly is always the best choice. Note that if shared in the stakeholders minds it may take differences due to the inherent differences of thought we have as human beings.

Comprehension and understandability of conceptual schemas and their components are the main object of research of this thesis. Although we can anticipate that conceptual schemas should contain a structural (sub)schema and a behavioral (sub)schema, we will describe its contents in detail in following chapters.

1.2 Large Conceptual Schemas

Nowadays the need of information that organizations have has clearly increased. As the amount of knowledge of an information system of an organization grows, the size of its conceptual schema also gets bigger following the same proportion.

As it is not the same programming in the large than programming in the small [18], likewise, to deal with small (sometimes know as *toy*) schemas is different than to specify, comprehend and study large conceptual schemas.

A complex and large conceptual schema is difficult to comprehend, limiting the accessibility to a small number of people, who have spent a significant amount of time understanding the schema. Obviously, the possibility that all the stakeholders of an information system spend that time is not acceptable.

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. These methods generate indexed, clustered, filtered, summarized or focused schemas that are easier to visualize and to understand.

To provide a conceptual schema with the most relevant knowledge highlighted implies a rise of knowledge accessibility that benefits the understandability of the schema. Otherwise we have a case of information overload.

1.3 Aim of this Thesis

As explained in the previous sections, to deal with large conceptual schemas is not an easy task for those who are not the modelers of the schema. In those cases, to give a reduced, less complex and understandable schema is mandatory if we want to increase efficiency.

There exists some alternative methods to process raw conceptual schemas and produce the desired output consisting in a simpler version of the input. Many of the above methods require computing the importance (also called relevance or, simply, score) of each element in the schema. The computed importance induces an ordering of the elements, which plays a key role in the steps and result output.

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

As far as we know, the existing metrics for elements importance are mainly based on the amount of knowledge defined in the schema, but only take into account a little part (concretely, a subset of the structural subschema) of it to compute that importance. Surprisingly, none of the methods we are aware of take into account all the knowledge about elements defined in the whole schema that, according to the intuition, could have an effect on the importance. A definition of complete schema [40] and its contents will be introduced in the next chapters.

Concretely we have selected a representative set of methods of the literature and have described them. We have tested those methods with some conceptual schemas we have. Afterwards, we have introduced new measures that consider the extra knowledge of the schema traditionally forgotten in the importance computation process. These measures induce extended versions of the base methods in the literature.

As before, we will define such extension approaches and test them with the same conceptual schemas. The results obtained are compared with the raw methods and some conclusions extracted.

To sum up, our intention here is to discover whether taking into account the

whole knowledge of the schema has an impact in the importance of the elements. Besides, we show some applications of the importance of elements in the process of increase the comprehension and accessibility of large conceptual schemas.

1.4 Outline of the document

Chapter 2 introduces some basics about the different areas of the thesis. Formal definition of conceptual schemas and its contents are presented, with some points about modeling languages, information retrieval concepts and the human perception. All of these are required knowledge to easily understand the following chapters.

Chapter 3 reviews the main contributions in the literature related to the topics of the thesis. Concretely, it contains descriptions of works on clustering of conceptual schema elements, a brief summary of filtering (or ranking) techniques, and some basics about visualization methods in the field of graph theory applied to conceptual modeling diagrams.

Chapter 4 presents the methods to compute the importance of entity types selected from the literature (some of them reviewed in Chapter 3). Furthermore, the metrics used by such methods are also included in the study, and some examples of application are described in order to increase understandability. Besides, such methods are extended with new metrics that take into account components of the conceptual schema traditionally forgotten.

Chapter 5 does an experimental evaluation of the methods presented in Chapter 4. Concretely, it includes a comparison between the similarity of the results obtained by both original and extended versions of the methods. Furthermore, the execution time is also evaluated and some advice are extracted in order to select the best method in any case.

Chapter 6 describes the implementation of the methods presented in Chapter 4 and evaluated in Chapter 5 into an existing modeling environment. It also includes the architecture selected and the way some of the methods must be computed in order to reach the solution. Finally, a brief user manual about the extended version of the environment is shown for the sake of understandability.

Chapter 7 concludes the thesis with a short review of the conclusions extracted from the explained research. In addition to it, a presentation of some notes about future work that must be done in the scope of the topic of relevance computing on conceptual modeling finishes the document.

Chapter 2

Background

This chapter starts with a description about the contents of complete conceptual schemas taking into account both structural and behavioural (sub)schemas in Section 2.1. Furthermore, Section 2.2 presents some modeling languages used to represent conceptual schemas. Concretely the aim of such section is centered in the UML and OCL as the modeling languages selected in this thesis.

The second part of the chapter continues in Section 2.3 with some basics about information retrieval and data mining areas which are becoming popular since many years ago due to the increasing size of information to manage. Finally, Section 2.4 justifies the need of methods and techniques to deal with huge amounts of information because of the limited capacity for processing information human beings have.

2.1 Conceptual Schemas

As introduced in Chapter 1, conceptual schemas are one of the key artifacts in the software engineering process. To be complete, a conceptual schema should contain two subschemas: the structural schema and the behavioural schema.

A conceptual schema contains the abstraction of the concepts of a concrete real-world domain. More precisely, each concept that has an important role in an organization and worths to be included in the information system is represented as an entity type.

Look at the example in Fig. 2.1. Imagine we have a domain of an organization that manages information about customers. This way, the concept *customer* which is important for the organization is represented in the information system by the entity type *Customer*. Therefore, real-world customers are represented by instances of the *Customer* entity type in the information system of the organization.

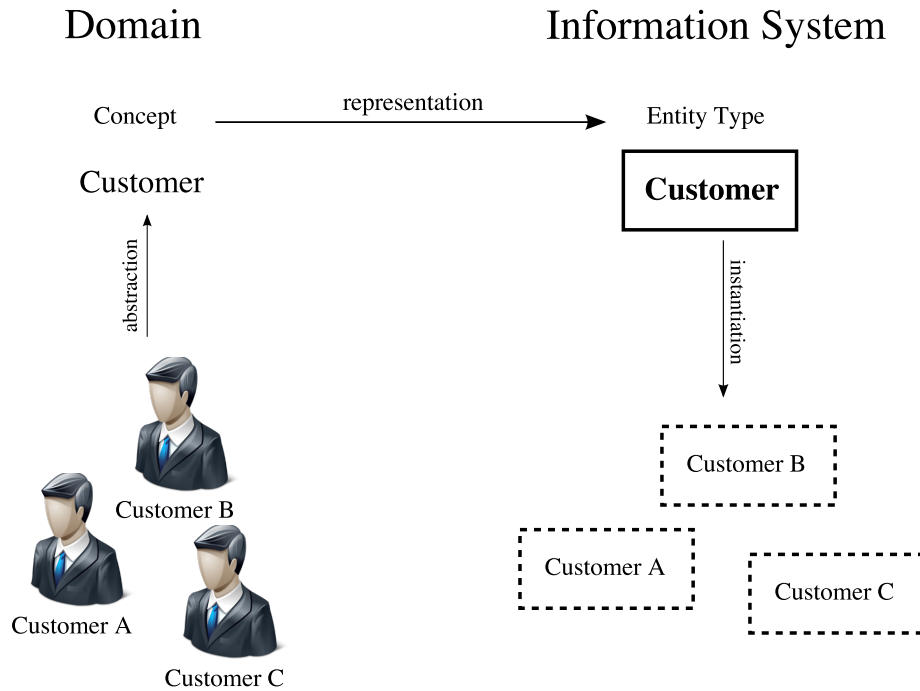


Figure 2.1: Classification of domain concepts into entity types.

Entity types are not independent. It is possible to have generalization / specialization relationships between any pair of them. In the previous example, the organization may have a special kind of customers, say gold customers. To denote this behaviour in the conceptual schema, it is possible to indicate that exists another entity type called *GoldCustomer* such that $IsA(Customer, GoldCustomer)$. It means that *GoldCustomer* is a specialization of the entity type *Customer*, and that *Customer* is a generalization of *GoldCustomer*. Each information about *Customer* in the information system is inherited by the *GoldCustomer* entity type.

Another element in conceptual schemas are relationship types. Each relationship type represent a connection or association between two or more entity types. In the previous example, imagine it is important to maintain information about the parents of a customer to, for example, offer them special discounts. The conceptual schema should contain a new relationship type $IsParentOf(parent:Customer, child:Customer)$. It means that exists a bidirectional reflexive relationship in *Customer* to denote the parents/children of an instance of *Customer*. The tags before the names of the entity types in the relationship represent the role names of such entity types as participants in the relationship.

The relationship types also have cardinalities. In the relationship $IsParentOf$, the cardinalities should be something like $Card(IsParentOf; parent, child) = (0, \infty)$ and $Card(IsParentOf; child, parent) = (0, 2)$. This way, an instance of *Customer* can have zero or more instances of *Customer* as its children; and an

instance of Customer can have at most two instances of Customer as its parents.

Another important characteristic to explain are Attributes. An attribute is a property of an entity type that contains information about it. Concretely, an attribute can be seen as a special relationship with two participants: an entity type and a data type. If the organization in our example wants to maintain the name of the customers in its information system, it is possible to declare an attribute in the form of $HasName(Customer, name : String)$. It means that an instance of Customer is related to an instance of the data type String (e.g. the word 'John C. Brown') that is the name of the customer. Data types are basic types and there exists some predefined like String, Integer, Real or Boolean.

The last concept we present are Schema Rules. A schema rule is a property about a subset of the conceptual schema that must be always satisfied. In our example about the management of customers, an instance of the entity type Customer could be related to itself through the association IsParentOf. This way, we could have that a customer is one of its parents. Obviously, this behaviour is not allowed and to denote it, we can specify a schema rule like:

$$Rule(NotParentOfItself : IsParentOf(c_1, c_2) \Rightarrow c_1 \neq c_2)$$

To sum up, we have briefly explained the main components of a conceptual schema. All extra information about conceptual schemas will be introduced in next chapters if necessary. For more about conceptual modeling take a look at Olive's book in [40]. Formally, we have:

Definition 2.1.1. (*Conceptual Schema*) A complete conceptual schema \mathcal{CS} is defined as a tuple $\mathcal{CS} = \langle \mathcal{SS}, \mathcal{BS}, \mathcal{E}, \mathcal{R}, \mathcal{G}, \mathcal{A}, \mathcal{SR} \rangle$, where:

- \mathcal{SS} is the Structural Schema. It will be explained in Section 2.1.1.
- \mathcal{BS} is the Behavioural Schema. It will be explained in Section 2.1.2.
- \mathcal{E} is a set of Entity Types. Each $e \in \mathcal{E}$ has a name and represents a concept in the domain of knowledge of the conceptual schema.
- \mathcal{R} is a set of Relationship Types. Each $r \in \mathcal{R}$ has a name and represents a relationship between entity types $e_1, \dots, e_k \in \mathcal{E}$. Note that the number of participants k in $r \in \mathcal{R}$ is the degree of r and $k > 1$.
- \mathcal{G} is a set of Generalization relationships. Each $g \in \mathcal{G}$ represents a directed relationship between a pair of entity types $(e_i \rightarrow e_j)$ where $e_i, e_j \in \mathcal{E}$ indicating that e_i is a direct descendant of e_j and e_j is a direct ascendant of e_i .
- \mathcal{A} is a set of Attributes. Each $a \in \mathcal{A}$ has a name and a type, and is owned by an entity type $e \in \mathcal{E}$.
- \mathcal{SR} is a set of Schema Rules. Each $sr \in \mathcal{SR}$ indicates an invariant, a derivation rule or a pre- or postcondition about a subset of the \mathcal{CS} .

2.1.1 Structural Schema

The structural schema is the part of the conceptual schema that consists on the set of entity and relationship types, as well as other elements that will be

mentioned below, used to observe the state of a domain. This part is also known as the static component of the whole knowledge of the information system.

As we can see in the next definition, the structural schema contains a subset of the conceptual schema contents that represents the relevant concepts, associations, properties and rules that must be maintained by the information system. Formally, we have:

Definition 2.1.2. (*Structural Schema*) The structural schema \mathcal{SS} is defined as a tuple $\mathcal{SS} = \langle \mathcal{E}_{ss}, \mathcal{R}_{ss}, \mathcal{G}_{ss}, \mathcal{A}_{ss}, \mathcal{SR}_{ss} \rangle$, where:

- $\mathcal{E}_{ss} \subset \mathcal{E}$ is a set of Entity Types.
- $\mathcal{R}_{ss} \subset \mathcal{R}$ is a set of Relationship Types. Each $r \in \mathcal{R}_{ss}$ has a name and represents a relationship between entity types $e_1, \dots, e_k \in \mathcal{E}_{ss}$. Note that the number of participants k in $r \in \mathcal{R}_{ss}$ is the degree of r and $k > 1$.
- \mathcal{G}_{ss} is a set of Generalization relationships. Each $g \in \mathcal{G}_{ss}$ represents a directed relationship between a pair of entity types ($e_i \rightarrow e_j$) where $e_i, e_j \in \mathcal{E}_{ss}$ indicating that e_i is a direct descendant of e_j and e_j is a direct ascendant of e_i .
- $\mathcal{A}_{ss} \subset \mathcal{A}$ is a set of Attributes. Each $a \in \mathcal{A}_{ss}$ has a name and a type, and is owned by an entity type $e \in \mathcal{E}_{ss}$.
- $\mathcal{SR}_{ss} \subset \mathcal{SR}$ is a set of Schema Rules. Each $sr \in \mathcal{SR}_{ss}$ indicates an invariant or a derivation rule in the context of a subset of elements in \mathcal{SS} .

Not all the entities and relationships types used to model the state of a domain need to be represented in an information system. The conceptual schema of an information system describes only the portion of the whole domain conceptualization containing the entities and relationship types that are useful for the organization and are represented in the information system.

For example, we can have an organization whose domain contains the concepts of Product, Customer and Worker but the organization only wants to represent Product and Customer in the information system.

2.1.2 Behavioural Schema

As Olive states in [40], the behavioural schema represents the valid changes in the domain state, as well as the actions that the system can perform. Changes in the domain state are domain events, and a request to perform an action is an action request event. We represent such events as entity types following the same approach as in [42]. This way, the same techniques to compute the importance of entity types can be directly applied to compute the importance of events.

An example of action request event can be the action of sending a mail to the system customers once a new product is included in the information system, while an example of domain event type can be the registration of a new customer in the information system. Take a look at Fig. 2.2.

The behavioural schema can contain relationship types whose participants

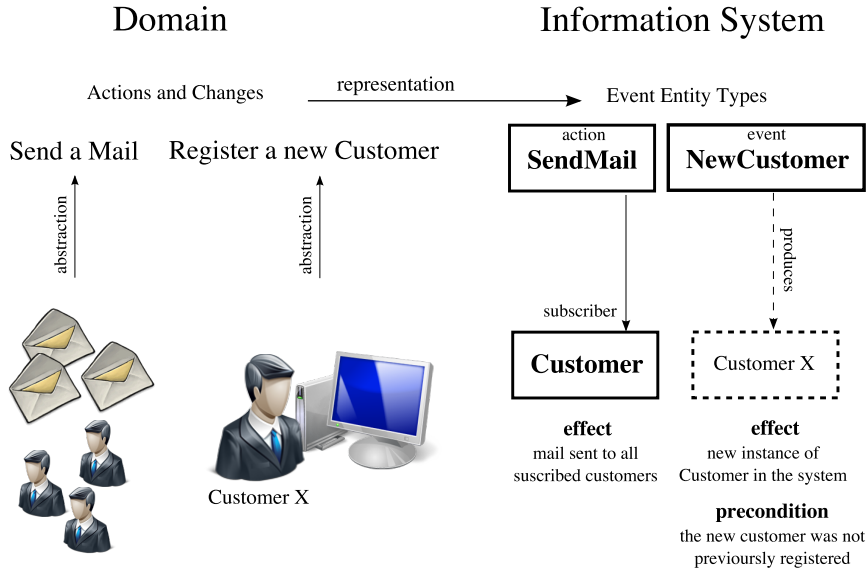


Figure 2.2: Example of event types.

include entity types from the structural schema related with entity types (event types) from the behavioural schema. Nevertheless, definition of structural entity types and relationship types between structural entity types are only placed in the structural schema.

Definition 2.1.3. (*Behavioural Schema*) The behavioural schema \mathcal{BS} is defined as a tuple $\mathcal{BS} = \langle \mathcal{E}_{bs}, \mathcal{R}_{bs}, \mathcal{G}_{bs}, \mathcal{A}_{bs}, \mathcal{SR}_{bs} \rangle$, where:

- $\mathcal{E}_{bs} \subset \mathcal{E}$ is a set of Entity Types. These are called Event Types.
- $\mathcal{R}_{bs} \subset \mathcal{R}$ is a set of Relationship Types. Each $r \in \mathcal{R}_{bs}$ has a name and represents a relationship between entity types $e_1, \dots, e_k \in (\mathcal{E}_{ss} \cup \mathcal{E}_{bs})$. Note that the number of participants k in $r \in \mathcal{R}_{bs}$ is the degree of r and $k > 1$.
- \mathcal{G} is a set of Generalization relationships. Each $g \in \mathcal{G}_{bs}$ represents a directed relationship between a pair of entity types ($e_i \rightarrow e_j$) where $e_i, e_j \in \mathcal{E}_{bs}$ indicating that e_i is a direct descendant of e_j and e_j is a direct ascendant of e_i .
- $\mathcal{A}_{bs} \subset \mathcal{A}$ is a set of Attributes. Each $a \in \mathcal{A}_{bs}$ has a name and a type, and is owned by an entity type $e \in \mathcal{E}_{bs}$.
- $\mathcal{SR}_{bs} \subset \mathcal{SR}$ is a set of Schema Rules. Each $sr \in \mathcal{SR}_{bs}$ indicates an invariant, a derivation rule or a pre- or postcondition in the context of a subset of elements in \mathcal{BS} . Such subset may also contain some elements in \mathcal{SS} .

The effect of an event type must be defined with pre- and postconditions included in the schema rules component of the behavioural schema. At bottom part of Fig. 2.2 there is an example of such definition for the domain event type NewCustomer.

2.2 Modeling Languages

A modeling language is an artificial language used to express information or knowledge about a domain. Modeling languages can be graphical or textual. Usually, graphical modeling languages are used to define the structure of concepts and its relationships using symbols and lines, while textual modeling languages are used to express what is not possible to express graphically like schema rules.

In the following subsections we briefly describe two popular modeling languages: the Unified Modeling Language –which is a graphical modeling language; and the Object Constraint Language –which is a textual modeling language. Such languages will be the used languages in the rest of this thesis.

Before the introduction of such languages, we make a short description about the Entity-Relationship model, which is the precursor of modern object-oriented approaches to model data and is broadly used in the literature about the topic of this thesis.

2.2.1 The Entity-Relationship Model

The Entity-Relationship (ER) model was firstly introduced by Chen in [14]. It defines a conceptual representation of data, formerly used for graphical database modeling.

The ER model introduces the concept of entity as an abstraction of some aspect of the real world which can be distinguished from other aspects of the real world. Furthermore it represents entities as rectangles and the relationships between them as diamonds, connected by lines to each of the entities in the relationship. Entities can be thought as nouns while relationships can be thought as verbs connecting two or more nouns. Finally, attributes are represented as ellipses connected to the entity or relationship that owns them.

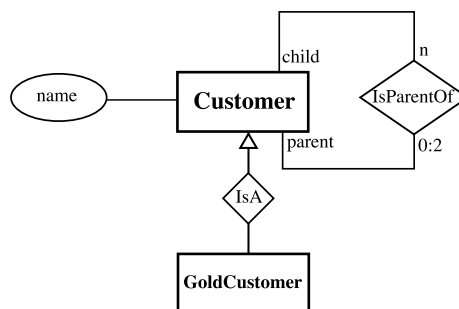


Figure 2.3: ER diagram about customers.

The graphical diagram containing entities, relationships and attributes is known as Entity-Relationship Diagram (or simply ERD). An example ERD

about customers explained in previous sections of the chapter can be found at Fig. 2.3. A more complex example from Wikipedia¹ is shown at Fig. 2.4.

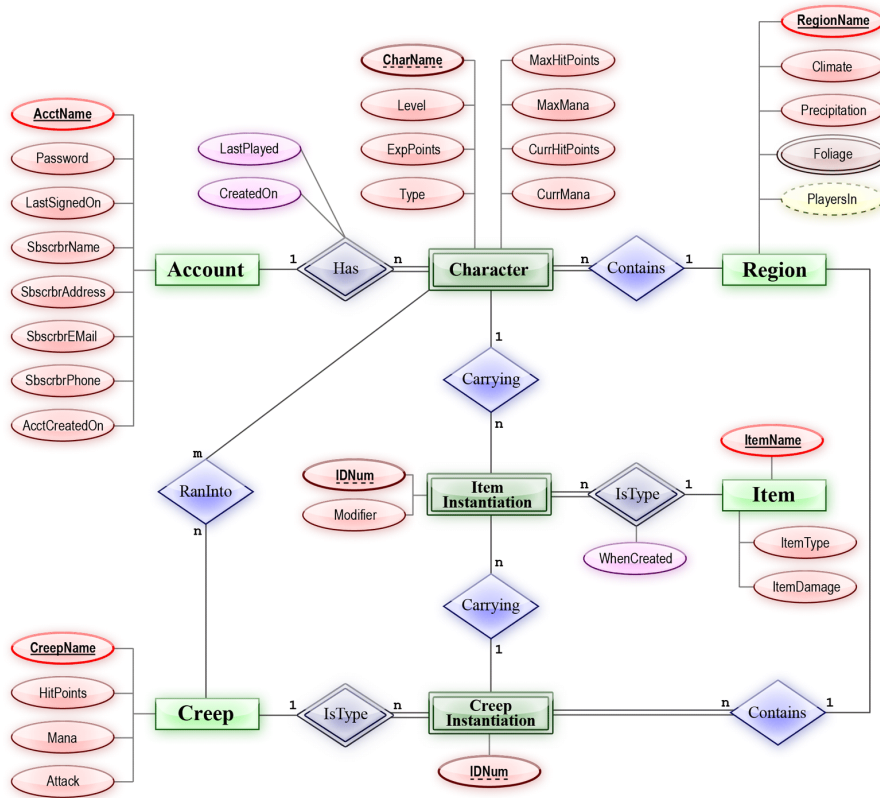


Figure 2.4: Entity-Relationship diagram.

The ER diagram notation has many variants and has evolved in the course of time. As we will see in the next chapter, many contributions in the literature about construct reduced, focused or filtered conceptual schemas are based on the ER notation and work with ER diagrams. Although in our thesis we focus on UML/OCL schemas, it is important to note that in the basis, both UML/OCL and ER schemas follow the same ideas and, therefore, the solutions to the problem of dealing with large and complex schemas for one of these modeling languages are valid solutions to the other type of modeling language.

2.2.2 The Unified Modeling Language

The Unified Modeling Language (UML) is a standardized general-purpose modeling language maintained by the Object Management Group(OMG). Its specification is public and actually the language is on version 2.0 [39].

¹http://en.wikipedia.org/wiki/File:ER_Diagram_MMORPG.png

The UML is a graphical language that contains several diagrams to specify a conceptual schema. An overview of the diagrams is depicted at Fig. 2.5. In this thesis we will only use the class diagram of the UML to define both the structural and behavioural subschemas of the conceptual schema. It is due to entity and relationship types can be defined as UML classes and associations. In the case of the event types of the behavioural schema, as we model them as events it is no necessary to use a special diagram different than the class diagram to specify them.

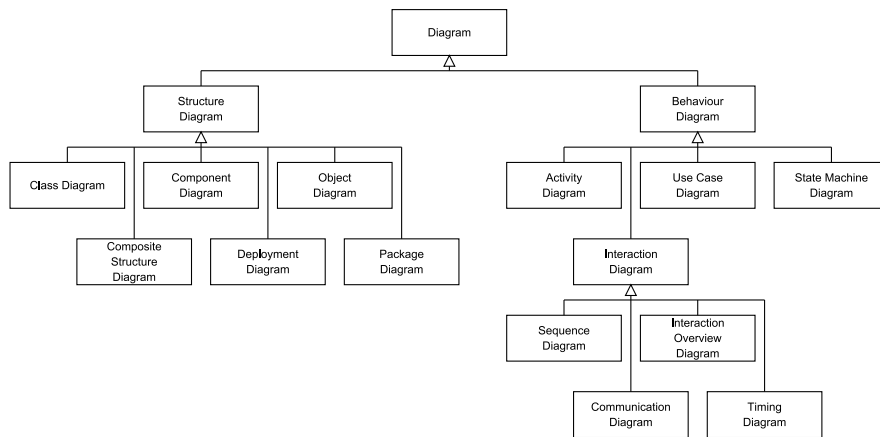


Figure 2.5: Class diagram of the 13 types of diagrams of the UML 2.0.

Modeling entity types as boxes and relationship types as links between them, the example of the conceptual schema about the management of customers is shown in Fig 2.6. Note that the attributes of each entity type is placed inside the attributes compartment of the related UML class. The cardinalities and roles of the relationship types are explicitly shown in the ends of each UML association between classes.

2.2.3 The Object Constraint Language

The Object Constraint Language (OCL) is a declarative language to formally describe rules in object-oriented models. The OCL is a widely accepted standard firstly introduced by IBM and now included in the UML and maintained by the Object Management Group (OMG). Its specification is public and actually the language is in the same version as UML (2.0)

The UML language provides a graphical notation based on diagrams to specify conceptual schemas as explained before. Nevertheless, invariants, derivation rules and pre- and postconditions must be expressed in natural language as comments in the diagrams of the schema. Thus, UML alone does not provide support for specify schema rules. Since the adoption of the OCL as part of the UML, it is possible to express schema rules using formal notation through

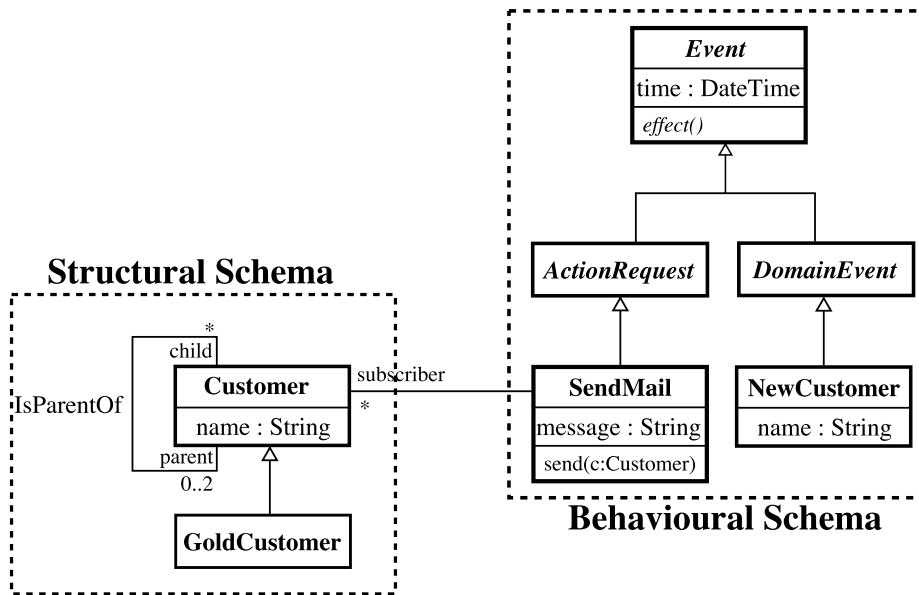


Figure 2.6: Example of conceptual schema specified in UML 2.0

OCL expressions, constructions and statements. A full explanation (a little bit outdated) about the syntax and semantics can be found in [57]. Another (more complex) source is [38], and a review of tools supporting OCL is described in [44].

In Fig. 2.7 there are the schema rules of our previous example about customers, including the invariants and pre- and postconditions of both structural and behavioural subschemas. We assume that the operation *send* of *SendMail* returns a boolean value according to if the message was sent successfully or not, because the body of the *forall* construction requires a boolean expression.

```

context Customer inv NotParentOfItself:
  self.parent->excludes(self)

context SendMail::effect()
  pre: message <> ''
  post: subscriber->forall(c:Customer|self.send(c))

context NewCustomer::effect()
  pre: Customer.allInstances()->forall(c:Customer|c.name <> self.name)
  post: cust.oclIsNew() and
    cust.oclIsTypeOf(Customer) and
    cust.name = self.name

```

Figure 2.7: Example of schema rules specified in OCL 2.0

2.3 Information Extraction

Nowadays information retrieval and, in a broader sense data mining, have become two of the most important disciplines to deal with large amounts of information. These subjects provide a set of different methods to extract knowledge from data. In-deep information about such topics can be found in [46] and [45]. More recent sources about information retrieval and data mining are [3] and [27], respectively.

Information retrieval was initially centered in searching information within documents until the birth of the web. As the Internet information became bigger, increasingly diffuse and complex to manage, information retrieval got a major prominence. One of the main contributions of this science was the appearance of the web searchers.

On the other hand, data mining has a broader scope of application. Some of its methods are used in statistics, programming and specially in data bases. As we will see in next chapter, literature contributions about reducing schemas have principally centered on database schemas.

Along this thesis, some of the main techniques will be used to compute the importance of schema elements (mainly entity types). Principally, link analysis and occurrence counting are the focus of our work.

Link analysis studies the edges of a connected graph to provide a ranking of those nodes that are more important according to its inner and outer connections through edges. This method is recursively defined and needs iterative algorithms to solve the problem. That is because the importance of a node comes from the other nodes that point to it and, therefore such importance flows to the nodes pointed by the node in question. This importance propagation must be computed in an equilibrium point where the graph nodes are balanced. Principally, such approach was introduced by Brin and Page as the foundation of Google's PageRank in [11].

Alternatively, occurrence counting is a basic technique that consists on to count how many times an element appears on a situation. It was centered on word occurrences in texts to discover the most/less used words or to state similarities between documents. As we will see in the next chapters, we apply this idea to conceptual schemas counting the number of occurrences of schema elements in different contexts.

Let's look an example of occurrence counting. Figure 2.8 contains a tag cloud with the top-150 words included in the superstructure specification document of the UML 2.0 language (see [39]). As the reader can see, a tag cloud is a cloud of words where the words with a greater number of occurrences have a greater size than the others.

Without read that document, it is possible to say that probably such document contains information about an specification, a superstructure, states, actions, elements, types, objects, classifiers, attributes, associations, constraints

2.4. Human Capacity for Processing Information

lems, like parallel computing or to have several replications of core processors. However, such solutions are not directly applicable to human beings.

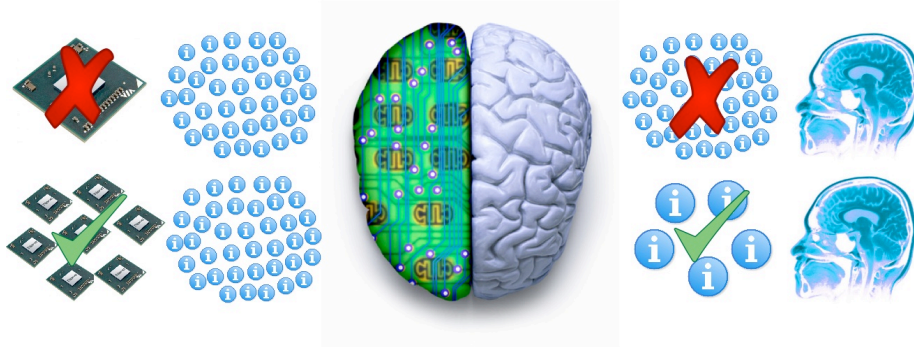


Figure 2.9: Bottleneck in human and computer capacity for processing information and solutions. While computers can be replicated, each human being only can have a brain, so the solution is to reduce the amount of information to process.

As we cannot replicate our brains or (generally) parallelize tasks, our purpose is to cut down the amount of information to process. Some solutions in this are are presented to reduce bottlenecks in human capacity for information processing, like clustering, that consists on group items according to some sort of similarity, or filtering, that hides irrelevant information, increasing the attention to important items. Such methods are highly recommendable in conceptual modeling to improve accessibility and comprehension of large conceptual schemas by reducing or changing the structure of their information.

Chapter 3

State of the Art

This chapter starts with a description about some notes taken from the literature that introduce the problem of large conceptual schemas management in Section 3.1. Furthermore, Section 3.2 explains the major contributions about schema clustering in the literature. As we will see there, this is a well explored area with a big amount of proposed solutions.

The second part of the chapter continues in Section 3.3 with some exploration about contributions in the filtering of conceptual schemas area. In this case, there is no agreement on the name of this topic, and it is possible to find synonym areas of filtering in the literature, like scoring or ranking.

Section 3.4 briefly describes some contributions defining metrics to measure some characteristics about conceptual schemas. Such metrics will be used (after a process of adjustment) in the main chapters of this thesis once introducing the methods we use to compute the importance of schema elements. And Section 3.5 shows some existing works dealing with how to visualize conceptual schemas in order to improve understandability.

Finally, Section 3.6 finalizes the chapter with some conclusions about the explanations introduced along the previous sections.

3.1 Requests for Contributions

In their research agenda for Conceptual Schema-Centric Development (CSCD) [41], Olivé and Cabot describe that we need methods, techniques and tools to support designers and users in the development, reuse, evolution, and understanding of large schemas. They dedicate a section of their agenda to explain that the development of large conceptual schemas pose specific problems not found in small conceptual schemas.

They also indicate that work on this topic has focused mainly on conceptual

schemas for databases and, as a conclusion, it is important to deal with information systems and to take into account both the structural (including constraints and derivation rules) and behavioural schemas. As explained in previous chapters, to follow such indications will be our purpose in computing the importance of schema elements.

Another request for contributions in this area is found in the study of Lindland, Sindre and Sølvsberg about quality in conceptual modeling [33]. They classify filtering as a modeling activity to improve the comprehension goal and model properties as structuredness, executability and expressive economy.

Concretely, they view filtering as a necessary activity because a person cannot grasp the entire schema at once and must concentrate on specific parts or aspects at a time. Filtering may also include aspects of translation because a large schema may have a rather diverse audience. Therefore, different languages will be preferred by various groups. While end users may want to see business rules in natural language, analysts may want to see them in logic.

This way, in addition to filtering they indicate that different views using different languages should be made in order to simplify the understandability of all kind of users of the conceptual schema.

Finally, Papazoglou claims in [43] that to improve the utility of large and complex information systems, we need to make schema interfaces more perceptive, responsive, and efficient for all categories of users including casual users. The author also requests the use of schema semantics instead of only the structure of the schema to reach this goal.

3.2 Clustering

Clustering can be defined as the activity of grouping elements according to a similarity function. Therefore, similar elements will be put together in the same group, or cluster. There exists a huge amount of contributions in the literature about clustering of schemas, ontologies or, definitely, graphs.

Estivill-Castro wonders in [19] why the existence of so many clustering algorithms. The answer here is clear: there are many clustering algorithms because there are many algorithms for each inductive principle and there are many inductive principles to solve the same problem. The author explains that clustering is in part in the eye of the beholder, meaning that every researcher can propose his own similarity function to approximate a solution. Because clustering is an optimization problem, the number of approximated solutions closer to the best solution is huge.

In this section we will review some solutions proposed to solve the problem of clustering the elements of conceptual schemas in the conceptual modeling area.

The paper from Feldman and Miller [20] is one of the foundation papers in such area. They explain the technique called entity model clustering and state that entity relationship diagrams can be manipulated in various ways to be more easily comprehended.

One of the problems the authors define is that the usefulness of any diagram is inversely proportional to the size of the model depicted. They consider any diagram with more than about 30 entity types to be reaching the limits of easy comprehension, depending on the number of relationships –the more relationships, the less comprehension is possible due to the accompanying increase in complexity. Therefore, it is possible to say that the two main problems of conceptual schemas are about size and complexity.

The entity model clustering technique proposed consists of a decomposition of the diagram in a tree with three levels of abstraction. Some entity types are allowed to be duplicate in some branches of the tree according to the authors' experience. Although the number of levels is determined by the diversity and complexity of an organization, they state that in practice three levels of diagram have been found to be useful. The hierarchy of successively more detailed entity relationship diagrams is shown in Fig. 3.1.

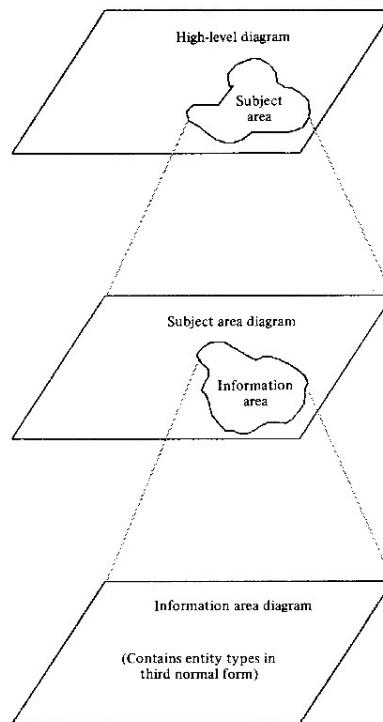


Figure 3.1: Three levels of abstraction diagramming. Extracted from [20].

First step consists on finding the major entity types. Occurrences of a major entity type should be uniquely identifiable from any related entity types. Fur-

thermore, a major entity type should be of fundamental importance to more than one functional area of organization, i.e. should appear in more than one information area.

Next step is to detect subject areas of the higher level. Subject areas and their information areas can be thought of as decompositions of the relationships between major entity types. Information areas are formed by first abstracting minor entities into a logical horizon and then successively abstracting the logical horizons and majority entity types. This process actually results in more than one information area relating to the same group of major entity types. These similar information areas are then abstracted to form a subject area, which is placed in the highest level.

Finally, Feldman and Miller classify the benefits of entity model clustering in:

- Organizational benefits: levels of diagrams are similar to levels in the organization.
- End-user computing benefits: they do not need to have to know of the existence of an entity type, but can be led to it through the succeeding levels of detail of the clustered entity model.
- Information system development benefits: development activities can be built without the complexity associated to large models.
- Entity modeling benefits: very large models become easy to communicate, validate and maintain.

Another foundational paper is the one from Teorey et al. [51]. Their approach is similar than Feldman and Miller's, but it contains some differences. The clustering method does not allow duplicate entities in different levels of the clustering hierarchy and the number of levels is not predefined.

First of all, the authors define four grouping techniques that make the same work than a similarity function:

- Dominance grouping: Dominant objects can be identified as the major entities. Each dominant object can then be grouped together with all its related (non-dominant) objects into a cluster.
- Abstraction grouping: Multi-level data objects using such abstractions as generalization, aggregation, classification, and membership (association) can be grouped into an entity cluster.
- Constraint grouping: Constraint-related objects that extend the ER model to incorporate the integrity constraints can be grouped into an entity cluster.
- Relationship grouping: The n-ary relationships of any degree can potentially be grouped into an entity cluster.

An example of dominance grouping operation is shown in Fig 3.2 where the entity type *Book* is clustered with its non-dominant connected entities. Note that the cluster name is the name of the major entity type and each cluster contains the number of level in the clustering hierarchy.

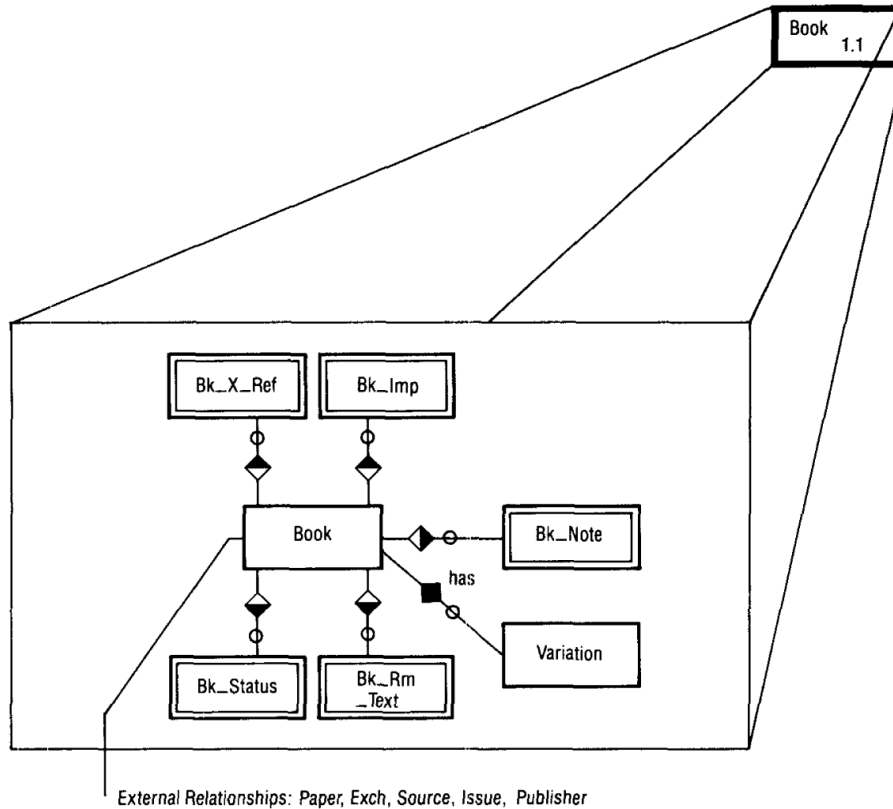


Figure 3.2: Example of dominance grouping operation. Extracted from [51].

Roughly, the clustering technique from Teorey et al. follows the next steps:

1. Define points of grouping within functional areas: locate the dominant entities in a functional area, either through the natural relationships as obtained from the system requirements document, local n-ary relationships, integrity constraints, abstractions, or by just being the central focus of many simple relationships.
2. Form entity clusters: use the basic grouping operations on elementary entities and their relationships to form higher-level objects, entity clusters.
3. Form higher-level entity clusters. Apply the grouping operations recursively to any combination of elementary entities and entity clusters to form new levels of entity clusters (higher level objects).

4. Validate the cluster diagram. Check for consistency of the interfaces (relationships) between objects at each level diagram. Verify the meaning of each level with the end users.

The result of applying these steps to an example of entity relationship diagram is shown in Fig. 3.3

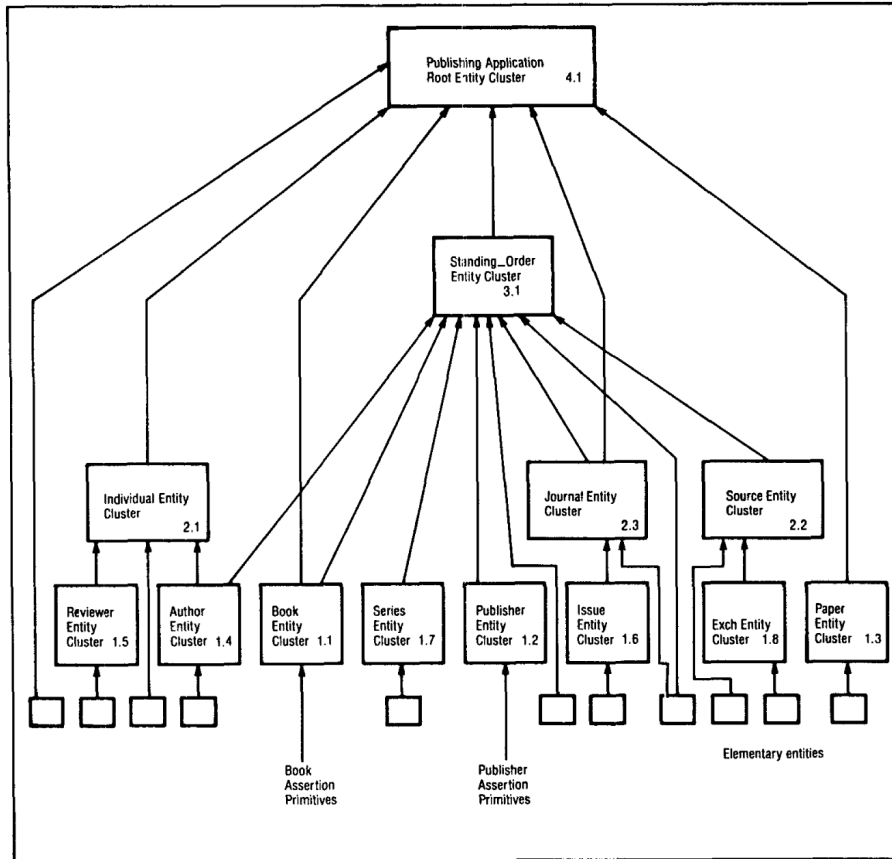


Figure 3.3: Example of Entity Cluster Levels. Extracted from [51].

Shoval, Danoch and Balabam in [48, 47] do a revision and refactors the approach of Teorey et al., previously defined. They call their new solution HERD (Hierarchical Entity-Relationship Diagrams) and basically includes minimum changes in grouping operations and their application.

Another different contribution was made by Jaeschke, Oberweis and Stucky [28]. In that paper the approaches to entity model clustering have been extended to allow top-down design. The main idea is to determine the major entity types and the coarse relationship types between them. Then these relationship types are refined iteratively by complex and simple relationship clustering, also involving entity clustering. After determining the major entity types, the de-

tailed design of the different relationship clusters can be realized simultaneously and independently by different project groups. This approach also supports database reengineering. The clusters can be built bottom-up based on already existing database schemes while the redesign is realized top down.

Roughly, the process consists on define high-level objects at first and, after that, redefine the relationships between them to complete the schema. An example of it is shown in Fig. 3.4.

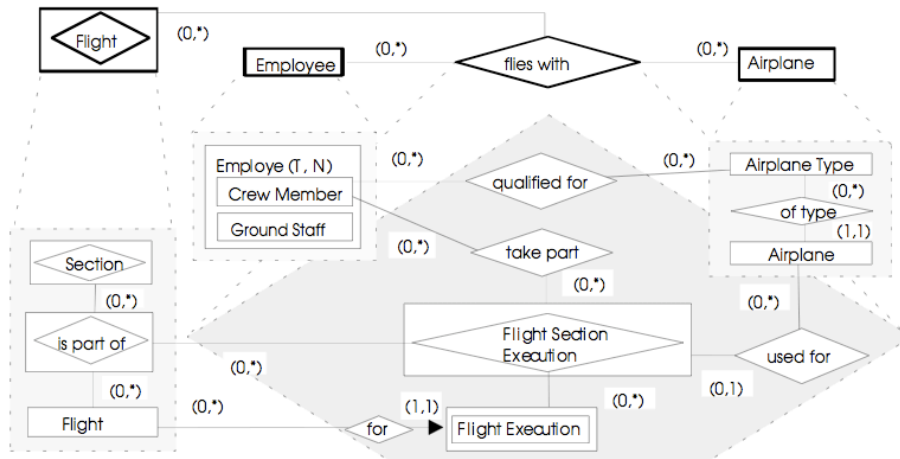


Figure 3.4: Example of Relationship Clustering. Extracted from [28].

Françalanci and Pernici in [21] discuss the problem of the semi-automatic construction of abstract ER schemas and propose an algorithm for schema clustering, mainly based on the structure of the schema. Furthermore, they define *affinity* and *closeness* between concepts and *coupling* and *cohesion* between clusters as the operating criteria for clustering:

- **Affinity:** captures semantic closeness among concepts. To compute it, they assign a list of words to each concept of a schema. The affinity between two concepts is calculated as the mean value of the similarities between each pair of words belonging to each concept.
- **Closeness:** corresponds to a quantitative evaluation of the links among concepts, i.e., an evaluation of syntactic closeness between concepts. Both the type and number of relationships between concepts suggest a strength of their closeness. Therefore closeness between two concepts is calculated as the addition of the strength for each relationship, according to its type, between the concepts.
- **Cohesion:** corresponds to a value indicating how well internally connected are the concepts within clusters.
- **Coupling:** corresponds to a value indicating the amount of connections between different clusters.

- Balance: the balance of a clustering of a schema can be measured as the standard deviation of the number of elements per cluster. It is interesting to have a clustered schema with clusters having similar number of concepts inside.

Another different approach for clustering is introduced in [22] by Gandhi, Robertson and Van Gucht. They define a new ER model called Leveled Entity Relationship (LER) model as a new way of modeling.

The *Leveled Entity Relationship* (LER) model is an adaptation of the traditional ER model adding a mechanism to make sub-entities deep inside an entity visible to the external world without unnecessary complexities. An LER entity may be atomic (like an ER entity) or it may have an internal structure. An aspect of an entity may be a direct property of the entity (like an ER attribute) or it may reflect an internal facet of the entity. And an LER relationship may directly associate two entities (like an ER relationship) or it may associate entities by linking subentities within them.

Look at Fig. 3.5 to see an example of LER model. Double arrowed edges represent the correspondence between aspects of the entity and aspects within the entity. Shaded boxes represent the self aspect of an entity and is used to define the internal structures of the entity. It is the focal point in the internal organization of the entity.

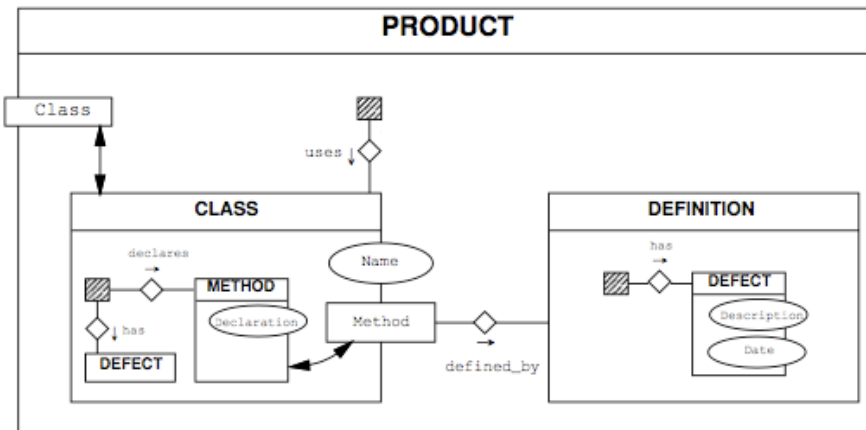


Figure 3.5: Example of Level Entity Relationship Model. Extracted from [22].

Next contribution that deserves a mention is Akoka and Comyn-Wattiau's paper [2] on ER and object-oriented automatic clustering. There, the authors propose a common clustering algorithm and a set of similarity functions that they call distances between elements. They define such distances and apply them in their algorithm. A short review of the distances, including object-oriented distances, is shown in the list as follows:

- Visual distance: two entities are said to be close if they are involved in the same relationship.
- Hierarchical distance: the distance between two entities is measured by the shortest relationship path between entities. The cardinalities of relationships are included in the computation of such distance.
- Cohesive distance: as with the preceding distance, the cohesive distance is measured by the shortest path between those entities. However, it is considered a different segment length and weight on such paths.
- Structural-connective distance: two elements are close if they are neighbors in a hierarchy (aggregation, generalization, whole-part structure). They are close if they are linked by an instance connection or a message connection. Otherwise the distance between two objects is the length of the shortest path between them.
- Category distance: two elements are considered to be very close if there exists a generalization relationship between them.
- Communicative distance: the communication between two objects expresses a semantic link between these objects. Therefore the communicative distance is based on message flowing.
- Frequent communicative distance: the message frequency is the number of times a message is flowing between objects for a given period of time. As a consequence, two objects are considered to be close when this frequency is high.

For their part, Campbell, Halpin and Proper formalizes in [12] a method for the strictly automatic selection of major entity (or object) types. Their approach sets apart from others because it considers the detailed conceptual semantics hidden in the constraints and also the manner in which the facts within the domain are verbalized. In particular, their approach utilizes the detailed constraint specifications and verbalizations provided by Object Role Modeling (a modeling technique that allows a variety of data constraints to be specified on the conceptual schema).

The semantics of these constraints allow to make the selection of major objects. The authors claim that their approach more accurately imitates human intuition than previous methods. As a second goal, the paper utilize the selected major object types in an algorithm to derive abstractions for a flat conceptual schema.

Using constraints to compute the major entity types is an approach followed by us in the next chapter of this thesis. We also take into account event types of the behavioural subschema.

Moody and Flitman in [37, 36] state that a Levelled Data Model consists of the following components:

- Context Data Model: a high level diagram, which provides an overview of the model and how it is divided into subject areas.
- Subject Area Data models: which show a subset of the data model in full detail. Foreign entities are used to show relationships to entities on other subject areas.
- Indexes: which are used to help locate individual objects (entities, relationships and attributes) within each subject area.

The model may be organized into any number of levels, depending on the size. Look at the example in Fig. 3.6.

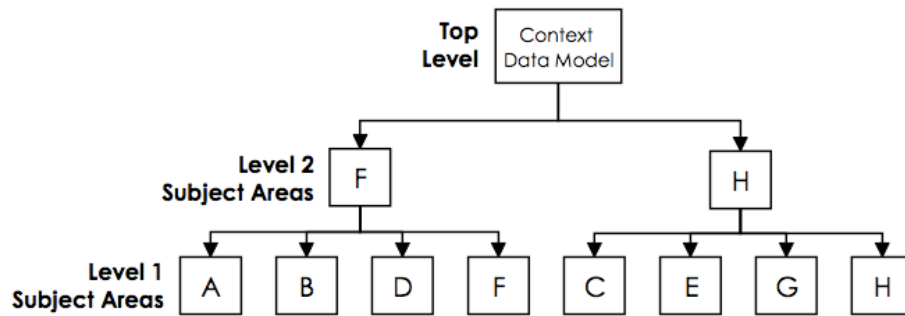


Figure 3.6: Example of Levelled Data Model. Extracted from [37, 36].

The authors also collect the major deficiencies identified in the existing literature:

- Lack of cognitive justification: to be truly effective in improving human understanding, clustering approaches need to be soundly based on principles of human information processing.
- Lack of size constraints: the aim of all existing methods is to reduce the model to parts of manageable size, but none of them define what this is.
- Lack of automation: only some approaches provide automated solutions to the problem.
- Levels of decomposition: most of the approaches proposed are limited to a fixed number of levels of decomposition.
- Lack of empirical testing: it is stated and argued by the authors that their methods provide an effective solution to the problem, but these claims are unsubstantiated.

Another good contribution by Moody and Flitman is the definition of formal rules or principles for evaluating the quality of decompositions and choosing between alternatives. These principles are briefly reviewed in the list as follows:

1. Completeness: each entity type must be assigned to at least one subject area. The union of the subject areas cover all the entities in the underlying model.
2. Non-Redundancy: each entity type must be assigned to at most one subject area. This ensures that subject areas are disjoint.
3. Integration: each subject area forms a fully connected subgraph of the original model.
4. Unity: each subject area should be named after one of the entities on the subject area, called the central entity. The central entity forms the core of the subject area.
5. Cognitively Manageable: each subject are must be of cognitively manageable size. It means a maximum of nine concepts (remember the *seven plus or minus two* number).
6. Flexibility: the partitioning of the data model into subject areas should allow adequate capacity for growth. A data model which consists of subject areas that are all of the size of nine will have to be repartitioned if even a single entity is added. To solve this situation, it is required that the average size of subject areas is as close as possible to seven entities.
7. Equal abstraction: all subject areas should be similar in size.
8. Coupling: the number of relationships between entities from different subject areas should be minimum.
9. Cohesion: the number of relationships between entities on the same subject area should be maximum.

The last contribution of Moody and Flitman is the definition of a manual decomposition procedure (see Fig. 3.7) and an automatic decomposition procedure (using genetic algorithms) of conceptual schemas.

To conclude the section, Tavana, Joglekar and Redmond study in [50] the decomposition principles of Moody and Flitman previously described. Concretely, they made a test with experts to decide the importance of each principle denoted by an order of significance. Afterwards, the obtained experience allowed them to refactor the Moody and Flitman's principles into a reduced set of seven:

1. Semantically Meaningful: people familiar with the task domain should find the clusters logical and coherent.
2. Completeness: decomposition should cover all of the entities and relationships in the complete model and no entities or relationships should be left out
3. Non-Redundancy: each entity and relationship should be in one, and only one, cluster.

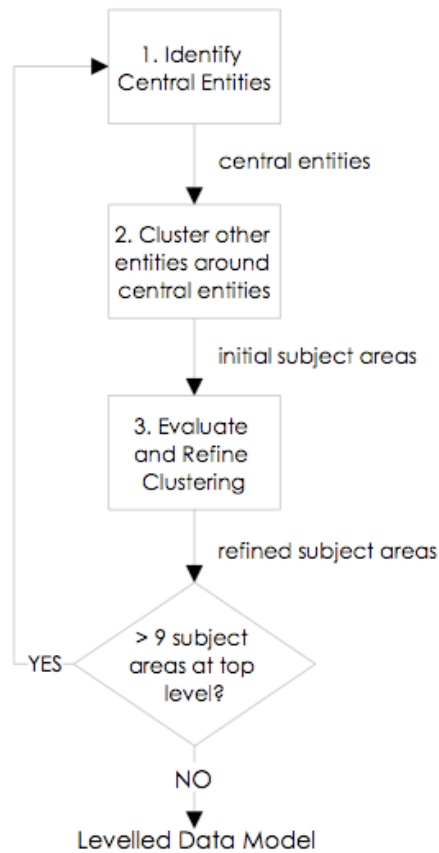


Figure 3.7: Manual Decomposition Procedure. Extracted from [37, 36].

4. Fully Connected: all the entities in a cluster should be connected to each other, via relationship paths that are within the cluster.
5. Maximal Cohesion Within Clusters: to the extent possible, all entities within a cluster should be closely related to each other.
6. Minimal Coupling Between Clusters: to the extent possible, entities in different clusters should not be closely related to each other.
7. High Degree of Modularity: provided all of the other criteria are satisfied, a solution with a greater number of clusters is preferred to a solution with smaller number of clusters.

The last part of the paper introduces a clustering algorithm that adopts some concepts and metrics from machine-part clustering in cellular manufacturing while exploiting some of the characteristics of ER diagrams. The aim of the algorithm is to follow the previous principles and, mainly, to reduce coupling and increase cohesion.

In fact, Tavana, Joglekar and Redmond make a comparison between their algorithm and the solutions proposed by Feldman and Miller, and Moody and Flitman. In both cases, the results obtained by their algorithm seem to be better than others. An example of the application of such algorithm is shown in Fig. 3.8.

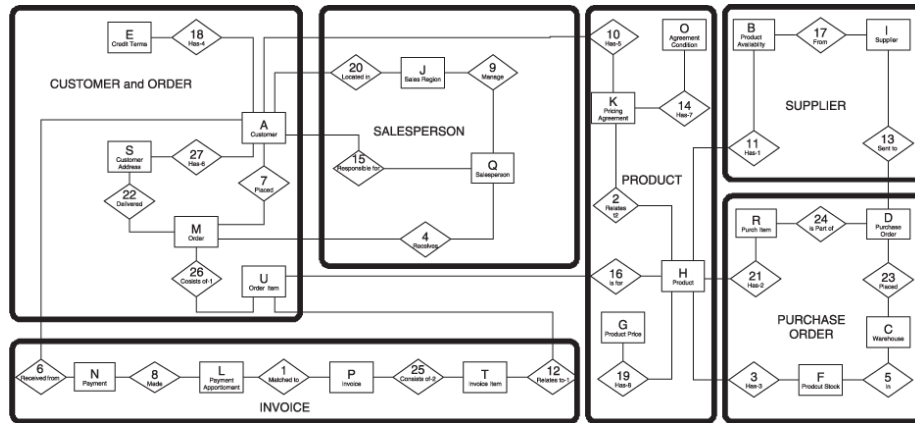


Figure 3.8: Application of clustering algorithm of Tavana, Joglekar and Redmond. Extracted from [50].

3.3 Filtering/Scoring/Ranking

In contrast to clustering of conceptual schemas, the number of research contributions on filtering (also known as scoring or ranking) methods applied to conceptual schemas have been clearly lower. In this section we review some of the most important papers in this area.

One of the first works on conceptual schema analysis was done by Castano, de Antonellis, Fugini and Pernici in [13]. The techniques proposed in the article include schema indexing methods, techniques for deriving abstract representations of large schemas and, similarity metrics for comparing schemas. Our interest about the article remains on the schema indexing method proposed, which will be explained in next chapter as one of the selected methods to compute the importance of entity types. However, let us take a look to this technique here.

Castano et al. state that the representative elements of a schema are its most relevant elements, that is, describing the purpose of the schema and its basic characteristics. Representative elements are determined on the basis of a *relevance* measure within the schema. To compute the relevance of an element, they take into account the properties of the element and the links in which it participates. The rationale is that the number of properties and links in which the element participates can be used as a (heuristic) measure of its relevance within the schema. The greater this measure, the higher the relevance of the

element, since this means that the element is characterized by several properties and is referred to by several other elements of the schema.

The method of Castano et al. only consider a small part of the structural subschema of a conceptual schema containing the entity types, their attributes, and the relationships between entity types (both association and generalization relationships). Roughly, for each entity type, its relevance is computed as the addition of the number of owned attributes plus the number of relationships in which the entity participates. In fact, each kind of characteristic (attribute, association, generalization) is weighted with a strength factor to denote a difference of the contribution of such kind in the final relevance value. The authors choose to give a higher strength to attributes, then generalizations and then association relationships. This is due to the fact that generalization/specialization links between entities in a hierarchy express a high connection between entity and its specializations, whereas relationships represent a weaker link, from the semantic point of view.

Concretely, to select the set of representative elements of a conceptual schema the steps are as follows:

- Computation of the relevance for each element (in this case it means entity type) according to the previous heuristic.
- Definition of a relevance threshold for selection of representative elements (e.g. the average relevance).
- Selection, as representative elements, such elements whose relevance exceeds the threshold.

Look at Fig. 3.9 to see an example where the bold squares are the representative elements computed according to the previous steps.

Also, Geerts, Mannila and Terzi adapt in [23] link analysis algorithms to relational databases. In analogy to link analysis algorithms, which use the Web graph to rank web pages, they use the database graph to rank partial tuples. To obtain rankings for partial tuples they mimic the principles of link analysis algorithms.

The well-studied algorithms for the Web show that the structure of the interconnections of web pages has lots of valuable information. For example, Kleinberg's HITS algorithm [30] suggests that each page should have a separate *authority* rating (based on the links going to the page) and *hub* rating (based on the links going from the page). The intuition behind the algorithm is that important hubs have links to important authorities and important authorities are linked by important hubs.

Brin and Page's PageRank algorithm [11], on the other hand, calculates globally the PageRank of a web page by considering a random walk on the Web graph and computing its stationary distribution.

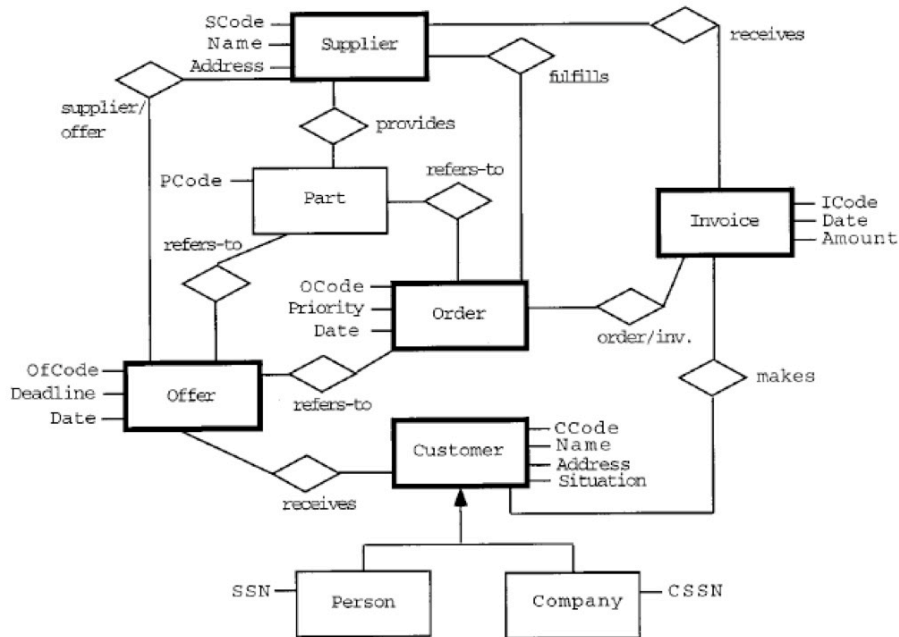


Figure 3.9: Representative elements of a conceptual schema. Extracted from [13].

The PageRank algorithm can also be seen as a model of a user's behavior where a hypothetical web surfer clicks on hyperlinks at random with no regard towards content. More specifically, when the random surfer is on a web page, the probability that he clicks on one hyperlink of the page depends solely on the number of outgoing links the latter has. However, sometimes the surfer gets bored and jumps to a random web page on the Web. The PageRank of a web page is the expected number of times the random surfer visits that page if he would click infinitely many times. Important web pages are ones which are visited very often by the random surfer.

Look at the example in Fig. 3.10. In such figure, size means relevance. It is easy to see that bigger (most relevant) smiley is the most pointed (or linked) one, and that the smileys linked by it get part of its importance becoming also big. That is due to the relevance flowing that link analysis algorithms produce according to their recursive definition: the relevance of an element is the addition of a proportional portion of the relevance of the elements that point to it. It is important to note that to execute such link analysis algorithms an iterative method is needed because of the recursive definition.

Geerts, Mannila and Terzi use this observation to extend the random surfer model to the random querier, which generalizes random-walk based link analysis algorithms by providing the random surfer with a different set of queries at his disposal. Additionally, the model facilitates extensions that allow for using this model for ranking partial tuples. The authors also apply HITS algorithm in the same way.

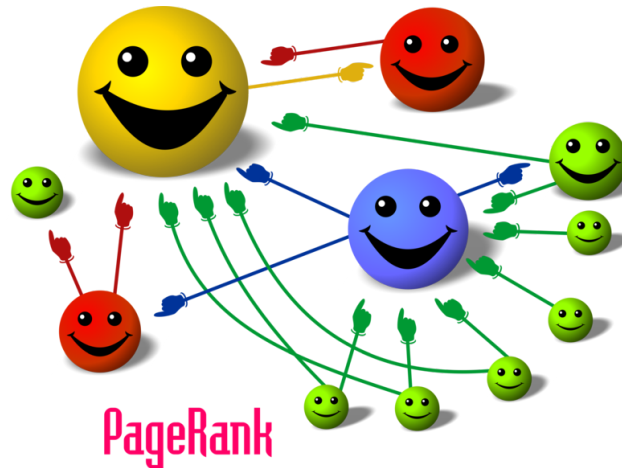


Figure 3.10: PageRank example.

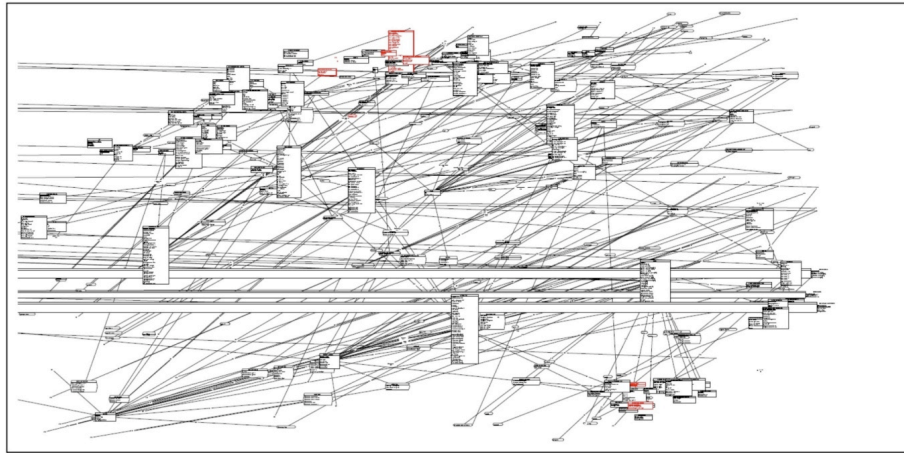
As they indicate, ranking tuples according to a query on a database is useful because to see the top- k partial tuples that satisfy a query rather than thousands of tuples ordered in a completely uninformative way increases database manageability.

Although we are not interested in ranking tuples in a database, the same approach can be used to rank entity types in a conceptual schema. Concretely, Tzitzikas and Hainaut propose in [54] two PageRank style algorithms, called EntityRank and BEntityRank, to obtain a rank of entity types according to their relevance in ER diagrams. The same algorithms, among others, are also included in the paper by Tzitzikas, Kotzinos and Theoaris [55], but applied to RDF schemas.

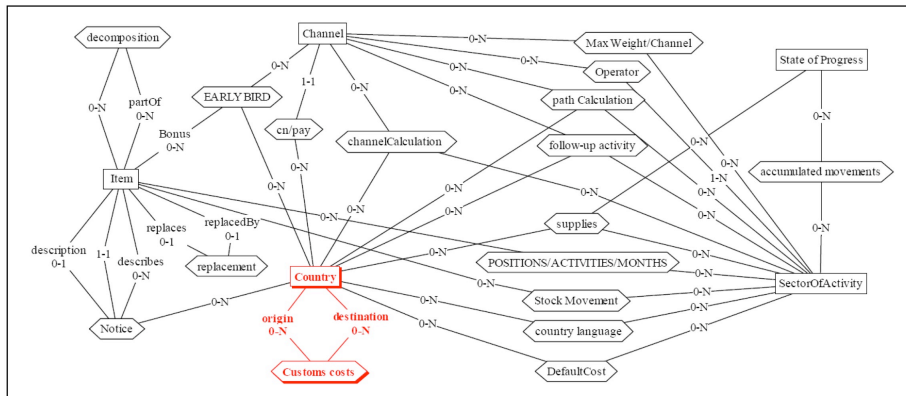
These two variants take into account in their computation only some structural elements of the conceptual schema. Concretely, to calculate the relevance of an entity type they use the relationships (without make a differentiation between generalization or association relationships). BEntityRank algorithm also assumes that the initial relevance of entity types is the number of attributes they own. After every iteration of the algorithms, the relevance gets closer to the real one due to the relevance transfers the algorithms produce through the relationships between entities.

EntityRank and BEntityRank are used to obtain the top- k entities in ER diagrams. Concretely, such algorithms produce a ranking that then is processed to filter those entities that are not the k most important ones. Look at the example of Fig. 3.11 to see a complete ER diagram (Fig. 3.11(a)) and the top-5 diagram (Fig. 3.11(b)) after the application of one of these algorithms.

A complete explanation with the definition of the algorithms introduced in [54] and [55], adapted to be used with conceptual schemas specified in UML/OCL, can be found in the next chapter.



(a) Full ER diagram



(b) Top-5 entity types

Figure 3.11: EntityRank application. Extracted from [54].

Last contribution we mention is the one by Yu and Jagadish [58]. It formally defines the concept of schema summary and two desirable properties (in addition to minimizing size) of a summary: presenting important schema elements and achieving broad information coverage. The method to obtain schema summaries is applied to database schemas.

A summary uses abstract elements and abstract links to summarize a complex schema and provide the users with a concise overview for better understanding. Each abstract element in the summary corresponds to a cluster of original schema elements (and other lower level abstract elements in the case of a multi-level summary), and each abstract link represents one or more links between the schema elements within those abstract elements. While schema summaries are useful, creating a good summary is a non-trivial task. A schema summary should be concise enough for users to comprehend, yet it needs to convey enough information for users to obtain a decent understanding of the

underlying schema and data.

Therefore, the importance of a schema element is reflected in two aspects - its connectivity in the schema and its cardinality in the database. The connectivity of an element in the schema graph provides a count of the number of other elements that are directly connected to it (via either relationship links). An important element is likely to be one from which many other elements can be reached easily. The cardinality of a schema element is the number of data nodes (database tuples) it corresponds to. If there are many data nodes of a schema element in the database, then that element is likely to be of greater importance than another one with very few data nodes.

The authors attempt to develop a single comprehensive notion of importance that combines these two aspects, and they use a PageRank-based algorithm according to the similarities of database schemas to the web. Unfortunately, we have studied the paper in deep and we do not agree with the affirmation of the more tuples a relation of the database has, the more important it is. We think in some examples in which this affirmation does not work. Imagine you are member of an organization that sells planes to a reduced group of exclusive customers from around the world. In your database probably there are a few instances of planes and customers, but if you maintain a list of the countries of the world where the customers come from, it could happen that the relation country have more tuples than plane or customer, but it is obvious that country is not more important than the others. Therefore, taking into account the number of tuples is not always a good idea.

Also, Yu and Jagadish use three relational database schemas to evaluate their algorithm - XMark, a database schema of an XML benchmark derived from an auction site; TPC-H, a relational benchmark for a decision support system; and MiMI, a real world scientific schema on protein interaction information.

They affirm that such schemas are large, concretely it is possible to find in the paper that XMark contains 327 schema elements, TPC-H contains 70, and MiMI 155. After our research on these schemas we have found that the real numbers for them are 29 for XMark, 9 for TPC-H and 38 for MiMI. Each number represents the number of relations (or tables) in each relational schema.

The difference between the number of elements is produced because Yu and Jagadish count the columns of the tables of the three relational database schemas also as elements, each one connected through a relationship with the element (the table or relation in the database) that owns it. We believe that it is not a good way to obtain a large schema for testing because, in this case, it is possible to obtain that an element, which in the real schema is a column of a table could be more important than the element that represents the table itself. Imagine we have a relation called Person to represent information about persons in our database, including three columns: id, name and phone. If we use *id* as primary key and it is used in other relations as foreign key, it could reach more importance than the relation Person. This behaviour is not acceptable and then the testing section of the article by Yu and Jagadish should be checked and corrected.

3.4 Metrics

The construction of similarity functions or heuristics to make clusters, or the computation of the importance of schema elements require to have a set of metrics that provide some measures taking into account the conceptual schema and, concretely both structural and behavioural subschemas.

Baroni et al. in [4, 5, 6, 7] present a library of measures, named FLAME (Formal Library for Aiding Metrics Extraction), which is mainly used to formalize metrics. Such library is formalized with OCL [38] upon the UML [39] metamodel.

Baroni's work includes the formalization of metrics of another libraries like MOOD (Metrics for Object-Oriented Design) [1] or MOOSE (Metrics for Object-Oriented Software) [15]. As we can see in Table 3.1, these are one of the metrics introduced by Baroni and used in the next chapters of this thesis (after a refinement process).

METRIC	DESCRIPTION
DAN	Defined Attributes in an entity type.
AAN	Available Attributes Number. Both defined and inherited (through generalization relationships) attributes of an entity type.
CHIN	Children Number. Directly descendant entities of an entity type (through generalization relationships).
PARN	Parents Number. Directly ascendant entities of an entity type (through generalization relationships).
DESN	Descendants Number. All descendants of an entity type (directed and undirected).
ASCN	Ascendants Number. All ascendants of an entity type (directed and undirected).
CN	Classes Number. Number of entity types in the schema.

Table 3.1: Some metrics introduced by Baroni et al. in [4, 5, 6, 7].

Genero, Poels and Piattini in [24, 25] defines a set of metrics for Entity-Relationship (ER) diagrams that can be used as indicators of the understandability of diagrams. Table 3.2 shows such metrics with their description.

The authors state that external qualities such as understandability and maintainability are hard to measure objectively early on in the modeling process. For a more objective assessment of external quality attributes, an indirect measurement based on internal model properties is required.

The metrics of Genero et al. allow compare different designs of the same domain in order to determine which one has a greater complexity or would be more difficult to maintain or comprehend. It is clear that a schema with greater

values for the metrics will be in a sense more complex and, therefore cause difficulties of understandability to end-users and other stakeholders.

In our approach we use such metrics (and others) to compute the importance of schema elements and, in particular, entity types.

METRIC	DESCRIPTION
NE	Number of Entities.
NA	Total number of Attributes in the ER diagram.
NDA	Total number of Derived Attributes in the ER diagram.
NCA	Total number of Composite Attributes in the ER diagram.
NMVA	Total number of Multi-valued Attributes in the ER diagram.
NR	Total number of Relationships.
NM:NR	Total number of Relationships with cardinality M:N.
N1:NR	Total number of Relationships with cardinality 1:N.
NN_AryR	Total number of N-ary Relationships.
NBinaryR	Total number of Binary Relationships.
NRefR	Total number of Reflexive Relationships.
NIS_AR	Total number of IsA (generalization) Relationships.

Table 3.2: Metrics introduced by Genero et al. in [25].

3.5 Visualization

Although visualization of large-sized conceptual schemas could be placed beyond the scope of this thesis, we cannot forget to present here some methods and techniques in the literature about this topic. The application of clustering or filtering methods must be followed by the application of visualization solutions in order to increase even more the understandability of schemas.

Tzitzikas and Hainaut explain in [53] that diagram drawing is not a panacea. It has been recognized long ago that the usefulness of conceptual diagrams degrades rapidly as they grow in size. Although the article focus in visualization of ontologies, it could be translated to database or conceptual modeling schemas written in ER or UML/OCL.

The authors include filtering and clustering as visualization techniques. That is a good classification because such techniques improve the graphical understandability of schemas. Furthermore, context-based browsing is also introduced. It consists on show only a short part of the whole schema so that the user could start browsing the diagram starting from any node of the schema. At each point in time, the neighborhood of the selected node is displayed. The user is then able to click on any other displayed node to change the focus. This way, the understanding of the schema is done gradually.

Another approach is presented by Streit, Pham and Brown in [49] to manage large business process specifications. Such specifications can be seen as graphs and, therefore, the application of the solution proposed be extended to conceptual schemas.

The solution here is adopted from the discipline of 3D computer graphics. It is possible to compare a large and complex diagram with a 3D representation of a full scale model. The authors explain that the purpose of simplification is to maintain a representation of the 3D model that is recognisable while reducing the processing and data requirements of the system. Similarly, in the case of conceptual schemas our purpose is to reduce the schema maintaining the relevant elements and a recognisable version of the whole while reducing the understandability effort users should made.

In Fig. 3.12 we can see the structure of a 3D model of a plane extracted from [49]. It is evident that the model continues looking as a plane although its detail level (complexity and size) is lower in the left side than in the right.

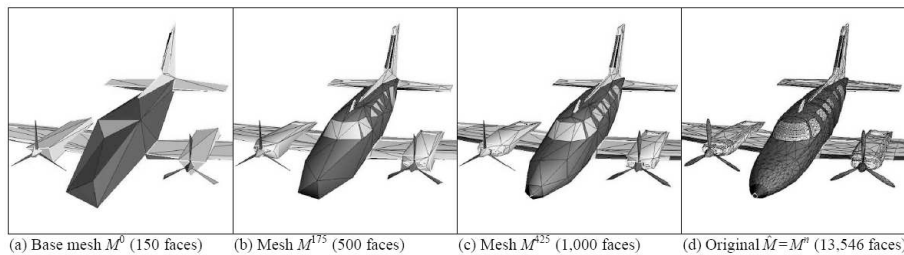


Figure 3.12: 3D mesh representing a plane at four different levels of detail. Extracted from [49].

The proposal of Streit, Pham and Brown consist on calculate the relevance of each node according to a criterion function. The second step is to reduce the graph (or schema in our case) by collapse or decimation methods. Finally, the graph is displayed for the user inspection. We can affirm that such solution is closer to filtering methods than to clustering.

Focus+Context is other visualization technique worths a mention here. Kosara, Miksch and Hauser in [32] explain how to use focus+context techniques in information visualization to point out relevant information to a user. They present a method for doing this which uses selective blur to direct the user's attention.

Roughly, the main idea is to blur objects based on their relevance. As human perception divides our field of view into foreground and background, most relevant objects must be placed closer (in the foreground) than less important ones (in the background). Look in Fig 3.13 to see an example of this technique to put the focus at persons in a picture hiding the details of the background.

Another contribution in the literature is the review of Cockburn, Karlson and Bederson [16]. It contains a summary of the different works and contributions in the visualization area and, in particular, those techniques classified in



Figure 3.13: Blur effect in Focus+Context technique to highlight persons.

Overview+Detail, Zooming, or Focus+Context methods.

Overview+Detail techniques are characterized by the simultaneous display of both an overview and detailed view of an information space, each in a distinct presentation space. The second category of interface supporting both focused and contextual views is based on Zooming, which involves a temporal separation between views. User magnify (zoom in) or demagnify (zoom out) a fragment of the information in the visualization area to focus on desired elements.

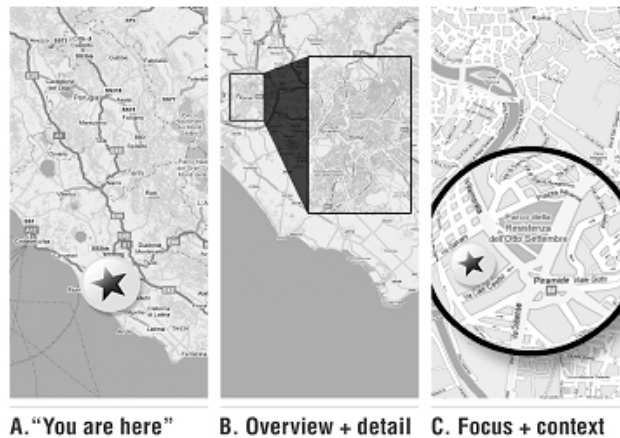


Figure 3.14: Example of visualization techniques.

Finally, Focus+Context as explained before, integrates focus and context into a single display where all parts are concurrently visible. The focus is displayed seamlessly within its surrounding context. An example of Focus+Context using the *fish-eye* technique is shown in Fig. 3.14(c).

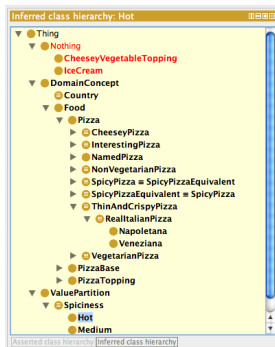
To finish this section, we also take into account the survey about ontology visualization methods of Katifori et al. [29]. Ontologies are sets of concepts and their relationships so that we can apply the methods explained in such survey to conceptual schemas.

The methods described in the survey are grouped in the following categories, according to their visualization type:

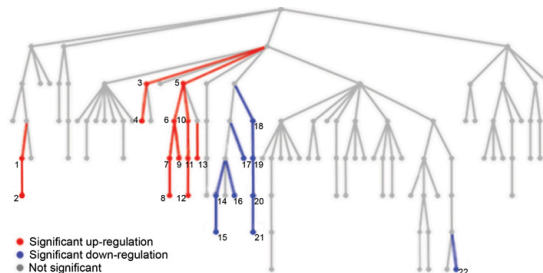
- Indented list
- Node-link and tree
- Zoomable
- Space-filling
- Focus+context or distortion
- 3D information landscapes

The Zoomable and Focus+Context categories are the same as in the survey of Cockburn et al. [16] previously explained.

On the other hand, the authors define Indented List techniques as lists where the elements of an ontology are hierarchically placed like in a directory tree view of common operating systems. Look at the example in Fig 3.15(a).



(a) Indented List



(b) Node-Link Tree

Figure 3.15: Example of Indented List and Node-Link Tree.

The next category is Node-link and tree, which represents ontologies as a set of interconnected nodes, presenting the taxonomy with a top-down or left to right layout. The user is generally allowed to expand and retract nodes and their subtrees, in order to adjust the detail of the information shown and avoid display clutter. Look at the example in Fig 3.15(b)¹.

¹www.gosurfer.org

3.5. Visualization

Then, Space-filling techniques are based on the concept of using the whole of the screen space by subdividing the space available for a node among its children. The size of each subdivision corresponds to a property of the node assigned to it—its size, number of contained nodes, and so on. Look at the example in Fig 3.16.

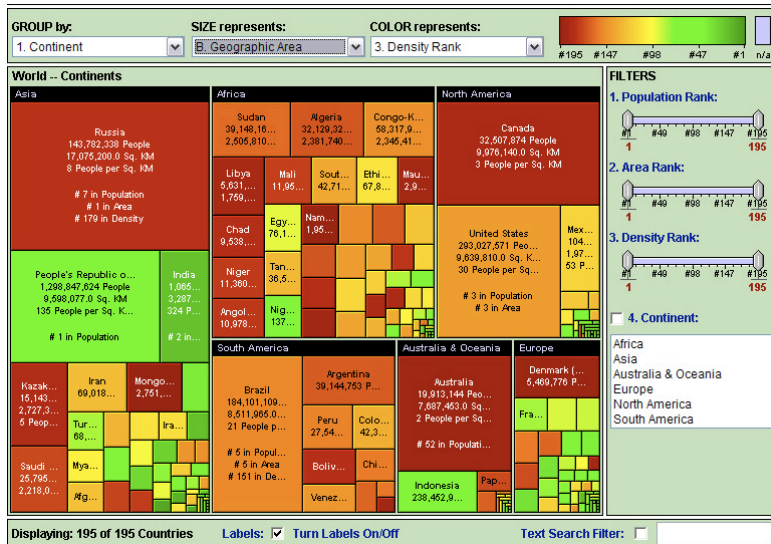


Figure 3.16: Example of World's Population visualized with a Tree Map, included in Space Filling techniques.

Finally, a very common metaphor used in Virtual Reality environments for document management is the landscape metaphor, where elements are placed on a plane as color- and size-coded 3D objects. Look at the example in Fig 3.17.

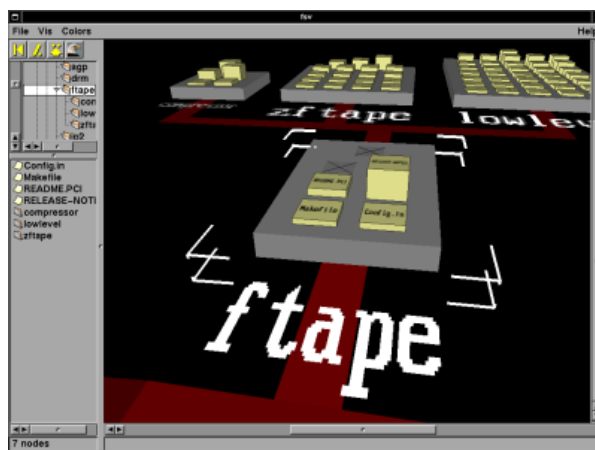


Figure 3.17: Example of 3D landscape of a Unix File System.

3.6 Conclusions

The main conclusions extracted from the previous sections are summarized in two affirmations we can construct due to the obtained experience.

On one hand, it is clear that the major contributions on dealing with the problems of conceptual modeling in the large in the literature are applied in the scope of databases or entity-relationship diagrams. As seen earlier in the chapter, although UML/OCL actually are the de facto standard modeling language, the number of methods to reduce or restructure conceptual schemas defined using UML/OCL is very small. Furthermore, a comparison between the huge amount of proposed solutions does not exist and makes difficult to select the best method for each situation.

On the other hand, most of the contributions do not take into account the whole knowledge provided by the conceptual schema. Concepts like constraints, derivation rules, definition of actions and events are commonly avoided. However, there exists the thought of the more knowledge used, the more complete the results will be.

Regarding these trends, we notice that a new approach to solve the problem of large conceptual schemas should be provided adapting and comparing existing methods to UML/OCL schemas, and including in their computation new measures, which could enclose the forgotten knowledge. These are the main points of this document.

Chapter 4

Measures, Methods and Extensions

In this chapter we briefly review the definition of some of the 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 adaptation to be applied to conceptual schemas modeled with UML/OCL.

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.

Section 4.1 introduces the measures (also called metrics) used in the base versions of the selected methods. Such measures are formally specified and an example is provided for the sake of understandability. Section 4.2 shows the selected methods of the literature and their definition.

Finally, Section 4.3 introduces new measures to take into account the whole knowledge of the conceptual schema and Section 4.4 presents the extension of the previous methods using such new measures. Section 4.5 concludes the chapter with a comparison of the methods and the knowledge used by both base and extended versions.

4.1 Measures

To cover the knowledge represented in a conceptual schema is not an easy job. There exists a set of modeling elements and a big amount of combinations between them to conform the structures and constructions that appear in the structural and behavioural sides of a conceptual schema.

In this section we adapt and define some of the main metrics in the literature

to gather the knowledge of schemas defined in UML. The main point here is the knowledge of the structural schema, since the great majority of methods focus on such part of the whole schema.

4.1.1 Importance-computing Principles

According to the experience obtained after the review of the state of the art in the field of how to compute the importance of schema elements, it is possible to say that most of the methods in the literature are based on subjective approaches.

A big number of solutions are evaluated comparing the results obtained by them and the solutions provided by an expert (or a set of them) in the information area where the conceptual schema defines the concepts of an information system. Such expert, also known as *oracle*, indicates the major elements in the schema, sometimes in a ranked list, according to its own experience.

This kind of evaluation that we can name as *subjective similarity* tends to a deviation produced by the inherent subjectivity of the validation. Roughly speaking, two *oracles* in the same area can have different points of view even if they work or have experience in the same aspects of the elements of a conceptual schema. Furthermore, to have the possibility of relying on an *oracle* that collaborate in the process of the application of a method to compute the most important schema elements is not possible in, at least, a big number of situations.

Thus, there exist the need of objective evaluations that do not change according to the influence of external factors like stakeholders of the information system. To reach this goal we present a set of objective metrics that only take into account the information represented in the conceptual schema. This way, the figure of the *oracle* can be avoided.

The first point is the definition of an axiom we follow along this thesis, and that have been studied in other research areas like text searching or document indexing in information retrieval or data mining. We call it the principle of *high appearance* of elements in a scope, and define it as follows:

Definition 4.1.1. (*Principle of High Appearance*)

The more occurrences an item/element has in an scope, the more important such item becomes.

According to the definition, the importance of an element has a proportional relation to the frequency with which such element appears in a scope. In the scope of conceptual schemas we redefine such principle for entity types:

Definition 4.1.2. (*Principle of High Appearance in Conceptual Schemas*)

The most important entity types of a conceptual schema are those that have the greater number of attributes, participate in a major number of associations,

have more ascendants and descendants, appear in the structure of schema rules, and definitely, have a bigger participation in the conceptual schema than others.

From the other point of view, the most important entity types are those that the designer of the conceptual schema requires a bigger effort and produces more information in the schema to define them.

This principle is directly related with the requirements engineering phase of software engineering. The requirements of an information system must be transformed into a conceptual schema using both the structural part and, mainly, the behavioural schema which can indicate the allowed actions and events of the system. Therefore, the requirements are converted into UML and OCL constructions.

Since the principle of *high appearance* induces that the most important elements are those that the conceptual schema has more information about, the relevance of such elements directly depends on the requirements. This way we have an objective approach based on schema metrics to denote the relevant elements, which has a component of subjectivity according to the information added by modelers and requirements engineers. Therefore, our approach is sufficiently objective to avoid deviations, but includes the required subjectivity to be useful.

4.1.2 Basic Metrics

Discuss about the relevance of schema elements has a point of ambiguity since there are a lot of types of elements in a conceptual schema. Although to compute the relevance of attributes, associations or constraints could be a research topic with interest, our approach, similarly as major contributions in the literature, has the computation of the relevance of entity types as the main goal of the thesis.

Obviously, the most important kind of elements in a conceptual schema are entity types, because they are the representation of concepts of the real world. Therefore, the application of methods for focus on the most relevant ones must contribute in a higher degree to increase understandability and usability.

In this section we go over the main concepts and the notation we have used to define the knowledge of conceptual schemas, as explained in Chapt. 2. In this thesis we deal with schemas written in the UML[39]/OCL[38]. Table 4.1 summarizes the notation (inspired by [55, 7]) used in the rest of the document.

As previously introduced, conceptual schema consists of a structural subschema and a behavioral subschema. The structural schema consists of a taxonomy of entity types (a set of entity types with their generalization/specialization relationships and the 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 (parent entity types) and descendants (children entity types) 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 $attr(e)$.

Notation	Definition
$par(e)$	$= \{e' \in \mathcal{E} \mid e \text{ IsA } e'\}$
$chi(e)$	$= \{e' \in \mathcal{E} \mid e' \text{ IsA } e\}$
$gen(e)$	$= par(e) \cup chi(e)$
$attr(e)$	$= \{a \in \mathcal{A} \mid entity(a) = e\}$
$members(r)$	$= \{e \in \mathcal{E} \mid e \text{ is a participant of } r\}$
$assoc(e)$	$= \{r \in \mathcal{R} \mid e \in members(r)\}$
$conn(e)$	$= \uplus_{r \in assoc(e)} \{members(r) \setminus \{e\}\}^1$
$par_{inh}(e)$	$= par(e) \cup \{par_{inh}(e') \mid e' \in par(e)\}$
$chi_{inh}(e)$	$= chi(e) \cup \{chi_{inh}(e') \mid e' \in chi(e)\}$
$attr_{inh}(e)$	$= attr(e) \cup \{attr_{inh}(e') \mid e' \in par(e)\}$
$assoc_{inh}(e)$	$= assoc(e) \uplus \{assoc(e') \mid e' \in par_{inh}(e)\}$
$conn_{inh}(e)$	$= conn(e) \uplus \{conn(e') \mid e' \in par_{inh}(e)\}$

Table 4.1: Definition of basic metrics.

The set of associations (relationship types) 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 $assoc(e)$, where $e \in \mathcal{E}$, 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 is the association `HasComponent(assembly:Part, component:Part)`, then $members(r) = \{\text{Part}, \text{Part}\}$, $assoc(\text{Part}) = \{\text{HasComponent}, \text{HasComponent}\}$ and $conn(\text{Part}) = \{\text{Part}\}$. Look at Fig. 4.1 to see the example.

The last row section in Table 4.1 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 . The other metrics capture the entire hierarchy of each entity type (e.g., $assoc_{inh}(e)$ collects all the associations where

¹Note that “ \setminus ” denotes the difference operation of multisets as in $\{a, a, b\} \setminus \{a\} = \{a, b\}$ and “ \uplus ” denotes the multiset (or bag) union that produces a multiset as in $\{a, b\} \uplus \{a\} = \{a, a, b\}$

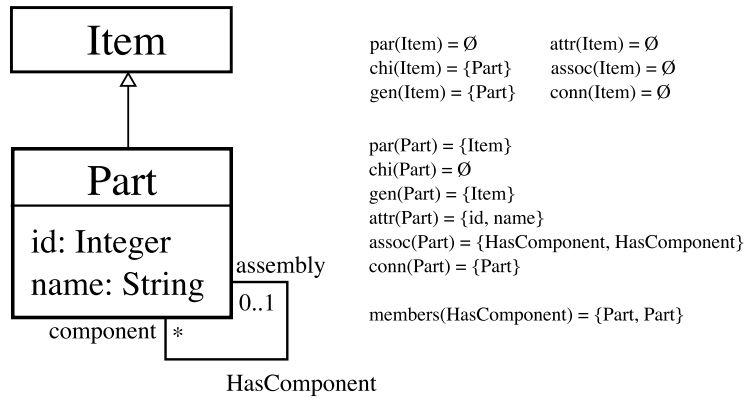


Figure 4.1: Example of basic metrics.

$e \in \mathcal{E}$ and its parents (and the parents of them, recursively) participates —as $\text{assoc}(e)$ is a multiset, $\text{assoc}_{inh}(e)$ can also contain repeated elements).

The set of basic metrics of Table 4.1 does not include metrics to gather the knowledge contained in the behavioural schema nor the schema rules of the structural schema. The reason is that the methods we have selected to study (described in Section 4.2) do not use such information. The notation for these new metrics is introduced in Section 4.3 before the description of the extended versions of the basic methods.

4.2 Methods

This section presents the methods from the literature that were selected to be included in the study of this thesis. These methods are based on occurrence counting and on link analysis. A brief description was done at Chpt. 3 about the state of the art in the field of how to compute the importance of entity types.

As such methods were principally applied to ER schemas, the definition here have been adapted with minor changes to accomplish the needs of the UML modeling language. The knowledge provided by OCL will be included in the extended versions of the methods in next sections.

Our purpose here is to formally describe a subset of methods in order to extend them in next sections due to the knowledge they take into account is not complete. Thus, the next are the base versions of the methods, using the information gathered into the previously mentioned metrics —a subset of the structural schema including entity and relationship types, attributes and generalizations.

4.2.1 The Connectivity Counter (CC)

The first method we present was firstly introduced by Moody and Flitman in [37]. They suggest that central entities should be chosen as the entities of highest business importance —the *core* business concepts in the model. Of course, *business importance* is quite a subjective concept, and requires human judgment. However a useful heuristic for identifying central entities is to identify entities with the most relationships. Usually the most highly connected entities are also the most important entities from a business viewpoint.

Following these indications we define the Connectivity Counter method as the method that computes the importance of each entity type as the number of relationships it participates in. Formally:

$$ICC(e) = |assoc(e)|$$

Look at the schema shown in Fig. 4.2. There are six entity types representing information about a simple store with products, customers and more.

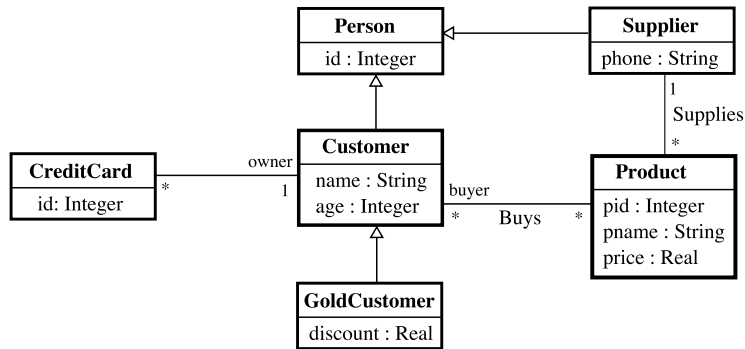


Figure 4.2: Example of schema.

The application of this method results in the values shown at Table 4.2 for the importance of the entity types presented in Fig. 4.2. Then, the most relevant entities are Customer and Product.

e	$ICC(e)$
CreditCard	1
Customer	2
GoldCustomer	0
Person	0
Product	2
Supplier	1

Table 4.2: Results for CC applied to example of Fig 4.2.

4.2.2 The Simple Method (SM)

This method was introduced in [55] (called m_0) and takes into account only the number of directly connected characteristics of each entity type. Formally, the importance $I_{SM}(e)$ of an entity type e is defined as:

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

Therefore, the importance of an entity type directly depends on the number of directly connected ascendants and descendants it has, the number of attributes it owns, and the number of participations in relationship types it does.

The values obtained after the application of the Simple Method to the previous schema are indicated in Table 4.3. There, it is possible to check that the most important entity type is Customer followed by Product.

e	$ par(e) $	$ chi(e) $	$ attr(e) $	$ assoc(e) $	$I_{SM}(e)$
CreditCard	0	0	1	1	2
Customer	1	1	2	2	6
GoldCustomer	1	0	1	0	2
Person	0	2	1	0	3
Product	0	0	3	2	5
Supplier	1	0	1	1	3

Table 4.3: Results for SM applied to example of Fig 4.2.

4.2.3 The Weighted Simple Method (WSM)

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 [13]. 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 $[0, 1]$. Concretely, we have selected the same strengths than the originals in [13]: $q_{attr} = 1$, $q_{inh} = 0.6$, and $q_{assoc} = 0.4$.

The results of the application of the weighted simple method to the previous example are shown in Table 4.4. The most relevant entity type is already Customer, followed by Product.

e	$q_{inh}(par(e) + chi(e))$	$q_{attr} attr(e) $	$q_{assoc} assoc(e) $	$I_{WSM}(e)$
CreditCard	0.6x0	1x1	0.4x1	1.4
Customer	0.6x2	1x2	0.4x2	4
GoldCustomer	0.6x1	1x1	0.4x0	1.6
Person	0.6x2	1x1	0.4x0	2.2
Product	0.6x0	1x3	0.4x2	3.8
Supplier	0.6x1	1x1	0.4x1	2

Table 4.4: Results for WSM applied to example of Fig 4.2.

4.2.4 The Transitive inheritance Method (TIM)

This is a variation of the simple method taking into account both directly defined features and inherited ones (see m_5 in [55]). 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:

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

As before, the results of the application of the TIM method to the previous example are shown in Table 4.5. Here, the most important entity type is GoldCustomer, followed by Customer and Product.

One of the problems of this method is that it gives more relevance to those entity types deeper in a hierarchy because such entities obtain the value (e.g., inherited attributes and participations in relationships) of their parents. That's why GoldCustomer is more relevant than Customer.

e	$ par_{inh}(e) $	$ chi_{inh}(e) $	$ attr_{inh}(e) $	$ assoc_{inh}(e) $	$I_{TIM}(e)$
CreditCard	0	0	1	1	2
Customer	1	1	3	2	7
GoldCustomer	2	0	4	2	8
Person	0	3	1	0	4
Product	0	0	3	2	5
Supplier	1	0	2	1	4

Table 4.5: Results for TIM applied to example of Fig 4.2.

4.2.5 EntityRank (ER)

The EntityRank method [54, 55] is based on link analysis following the same approach than Google's PageRank [11]. 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:

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

Such formulation produces a system of equations. Concretely, for the example of Fig. 4.2 such system would be as follows:

$$\begin{aligned} I_{ER}(\text{CreditCard}) &= \frac{q}{6} + (1 - q) \left(\frac{I_{ER}(\text{Customer})}{2} \right) \\ I_{ER}(\text{Customer}) &= \frac{q}{6} + (1 - q) \left(I_{ER}(\text{CreditCard}) + \frac{I_{ER}(\text{Product})}{2} \right) \\ I_{ER}(\text{GoldCustomer}) &= \frac{q}{6} \\ I_{ER}(\text{Person}) &= \frac{q}{6} \\ I_{ER}(\text{Product}) &= \frac{q}{6} + (1 - q) \left(\frac{I_{ER}(\text{Customer})}{2} + I_{ER}(\text{Supplier}) \right) \\ I_{ER}(\text{Supplier}) &= \frac{q}{6} + (1 - q) \left(\frac{I_{ER}(\text{Product})}{2} \right) \end{aligned}$$

It is important to note that the relevance of an entity comes from the relevance of the entities connected to it. The relevance flows through associations. For the case of Customer, its relevance come from CreditCard and from Product. As Product is connected with two entity types, a fragment of its relevance (the middle) goes to each of its connected entities (Customer and Supplier).

To compute the importance of the entity types we need to solve this equation system. If we fix that $\sum_{e \in \mathcal{E}} I_{ER}(e) = 1$ then the results obtained are shown in Table 4.6. We have chosen a $q = 0.15$, which is a common value in the literature.

e	$I_{ER}(e)$
CreditCard	0.16
Customer	0.30
GoldCustomer	0.04
Person	0.04
Product	0.30
Supplier	0.16

Table 4.6: Results for ER applied to example of Fig 4.2.

4.2.6 BEntityRank (BER)

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 [54, 55]. The higher the number of attributes, the higher the probability to randomly jump to that entity type. That is:

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

As in the case of EntityRank, the formulation produces a system of equations. In this case the system would be as follows:

$$\begin{aligned}
I_{BER}(CreditCard) &= q\frac{1}{9} + (1 - q) \left(\frac{I_{BER}(Customer)}{2} \right) \\
I_{BER}(Customer) &= q\frac{2}{9} + (1 - q) \left(I_{BER}(CreditCard) + \frac{I_{BER}(Product)}{2} \right) \\
I_{BER}(GoldCustomer) &= q\frac{1}{9} \\
I_{BER}(Person) &= q\frac{1}{9} \\
I_{BER}(Product) &= q\frac{3}{9} + (1 - q) \left(\frac{I_{BER}(Customer)}{2} + I_{BER}(Supplier) \right) \\
I_{BER}(Supplier) &= q\frac{1}{9} + (1 - q) \left(\frac{I_{BER}(Product)}{2} \right)
\end{aligned}$$

It is important to note that if an entity type is not connected to others through associations, its relevance only consists on its percentage of attributes multiplied by the coefficient of random jumps (q).

Similarly than with EntityRank, to compute the importance of the entity types we need to solve this equation system. We fix that $\sum_{e \in \mathcal{E}} I_{BER}(e) = 1$ then the results obtained are shown in Table 4.7. We have already chosen a $q = 0.15$, which is a common value in the literature.

One more time, the most important pair of entity types are Product and Customer. Therefore a focused view of the schema shown in Fig 4.2 could contain such two entities and the relationship type Buys placed between them.

e	$I_{BER}(e)$
CreditCard	0.15
Customer	0.31
GoldCustomer	0.02
Person	0.02
Product	0.33
Supplier	0.16

Table 4.7: Results for BER applied to example of Fig 4.2.

4.2.7 CEntityRank (CER)

Finally, the method that we call CEntityRank (m_4 in [55]) 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:

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

The formulation produces a system of equations as in the other two methods

based on link analysis. In this case the system would be as follows:

$$\begin{aligned}
I_{CER}(CreditCard) &= q_1 \frac{1}{9} + (1 - q_1 - q_2) \left(\frac{I_{CER}(Customer)}{2} \right) \\
I_{CER}(Customer) &= q_1 \frac{2}{9} + q_2 \left(\frac{I_{CER}(Person)}{2} + I_{CER}(GoldCustomer) \right) \\
&\quad + (1 - q_1 - q_2) \left(I_{CER}(CreditCard) + \frac{I_{CER}(Product)}{2} \right) \\
I_{CER}(GoldCustomer) &= q_1 \frac{1}{9} + q_2 \left(\frac{I_{CER}(Customer)}{2} \right) \\
I_{CER}(Person) &= q_1 \frac{1}{9} + q_2 \left(\frac{I_{CER}(Customer)}{2} + I_{CER}(Supplier) \right) \\
I_{CER}(Product) &= q_1 \frac{3}{9} + (1 - q_1 - q_2) \left(\frac{I_{BER}(Customer)}{2} + I_{BER}(Supplier) \right) \\
I_{CER}(Supplier) &= q_1 \frac{1}{9} + q_2 \left(\frac{I_{CER}(Person)}{2} \right) + (1 - q_1 - q_2) \left(\frac{I_{BER}(Product)}{2} \right)
\end{aligned}$$

Similarly than with EntityRank and BEntityRank, to compute the importance of the entity types we need to solve this equation system. We already fix that $\sum_{e \in \mathcal{E}} I_{CER}(e) = 1$ then the results obtained are shown in Table 4.8. We have chosen $q_1 = 0.1$ and $q_2 = 0.2$, which are some good values as indicated in [55].

As in the previous methods the most important pair of entity types are Product and Customer. However, we have obtained the similar rankings because the schema shown in Fig 4.2 is very small. The methods presented here must be tested with large schemas to bring out the differences between them.

e	$I_{CER}(e)$
CreditCard	0.13
Customer	0.32
GoldCustomer	0.05
Person	0.08
Product	0.27
Supplier	0.16

Table 4.8: Results for CER applied to example of Fig 4.2.

4.3 Extended Measures

This section presents some of the new approaches introduced in this thesis to gather the knowledge of the whole schema. First of all, we introduce how to deal with complex relationship types, including relationships whose degree is n ($n > 2$) and reified relationships.

Next step consists on describe the power OCL brings us to take into account the knowledge provided by the schema rules (\mathcal{SR}) of the conceptual schema. Our research here allows to uncover existing relationships between entity types placed in OCL expressions that are useful in the relevance computation process.

Finally, we mix the previous techniques to introduce new metrics to squeeze the knowledge previously extracted, mainly from the behavioural schema and

the schema rules of the structural schema. Such metrics will be the key stone in the extension of the base versions of the selected methods from the literature introduced in Section 4.2.

4.3.1 Complex Relationship Types

The Unified Modeling Language (UML) allows to specify a special kind of relationship types that can have the same behaviour as entity types: the association classes. Thus, the concept of *reification* is introduced as viewing a relationship as an entity. Reification can easily be defined in UML, as shown in left sides of Fig. 4.3 and Fig 4.4 for binary (entity C) and n -ary (entity AC) relationships.

As current methods in the literature are applied to schemas defined with ER diagrams but not with UML/OCL, the problem of how to take into account association classes is not explored in this field. To solve it, we propose to follow the same approach than Olivé in Chapt. 6 of [40]. It consists on implicitly reificate the current association class into another entity type associated through new binary relationships with its previous participants. Look at Fig. 4.3 to see the process.

It is important to maintain the same semantics before the explicit reification. Thus, the modeler has to take care of cardinality constraints. In the case of a binary reified relationship (left side of Fig. 4.3), the cardinality constraints after the explicit reification are interchanged between the ends of the relationship and the association class. This way, the cardinality constraints of an end go to the new relationship between the other end and the new entity representing the previous association class, in the side of such new entity type. In the side of the previous ends the cardinality equals 1.

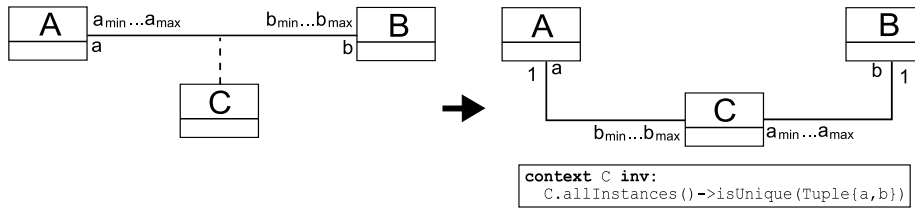


Figure 4.3: Implicit reification of a binary relationship with association class.

Furthermore, to maintain the semantics we also need a new uniqueness constraint for the new entity type after the implicit reification. Such constraint cannot be defined graphically, so the modeler requires the use of OCL. Such constraint can be written as shown in Fig. 4.3. Roughly, we need an invariant to check that the same pair of instances of both ends (A and B in the image) cannot be linked with the same instance of the new entity type (C in the example) more than once.

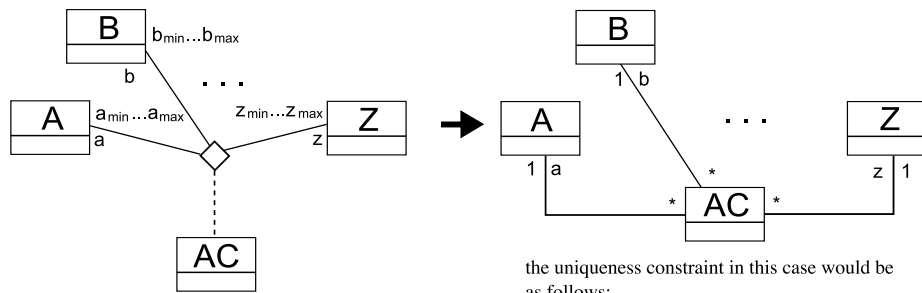
After this process, it is possible to use the same metrics previously defined

because the conceptual schema only contain common relationship types —the association classes have been converted into a group of binary relationships.

To sum up, in the case of binary relationships with association class the steps are as follows:

1. Define the entity type representing the association class.
2. Define the new binary relationship types with its cardinality constraints.
3. Define the uniqueness constraint.

Association classes in n -ary relationships need a different conversion process to be converted into explicit reifications. First step consisting on the conversion of the structure follows the same rules than in the case of binary association classes. Take a look at Fig. 4.4 to see that the association class (AC in the figure) is changed by a common entity type with new relationships reaching each of the initial ends.



the uniqueness constraint in this case would be as follows:

```
context AC inv:
  AC.allInstances()
  ->isUnique(Tuple{a, b, ..., z})
```

for each participant p with a $p_{min} > 0$, we need a new constraint like:

```
context AC inv:
  AC.allInstances()
  ->forall(ac1|AC.allInstances()
  ->select(ac2|ac1.a = ac2.a and
  ac1.b = ac2.b and
  ... and
  ac1.z = ac2.z
  )->size()>p_min
  )
```

where an equality condition appear for each participant in the body of the select instruction except for $ac1.p = ac2.p$

Idem for each participant p with a $p_{max} < *$. In this case, the comparison of the size() operation is done as follows: $...->size() < p_{max}$

Figure 4.4: Implicit reification of a n -ary ($n > 2$) relationship with association class.

After that, the new cardinality constraints (see right side of Fig. 4.4) are the same for each new relationship between the ends and the implicitly reified association class $1:Many$ from the ends to the new entity type.

As before, in the case of binary relationships with association class the steps are as follows:

1. Define the entity type representing the association class.
2. Define the new binary relationship types with its cardinality constraints.
3. Define the uniqueness constraint.
4. Define as constraints as needed to maintain the same cardinality constraints as before.

In this case a uniqueness constraint is also needed and follows the same idea than with binary relationships although this time the `Tuple` inside the OCL expression has as members the degree of the current association to reify —each one of the members is one of the ends. Furthermore, for every cardinality constraint in the initial phase (left side of Fig. 4.4) a new rule has to be introduced to maintain the semantics. Such rule must be defined in OCL following the same layout than the one shown in Fig. 4.4.

Another example of an implicit reification is shown in Fig. 4.5. We have a ternary relationship with the association class `Reservation`. Following the previous steps, `Reservation` is converted into a new entity type with three relationships.

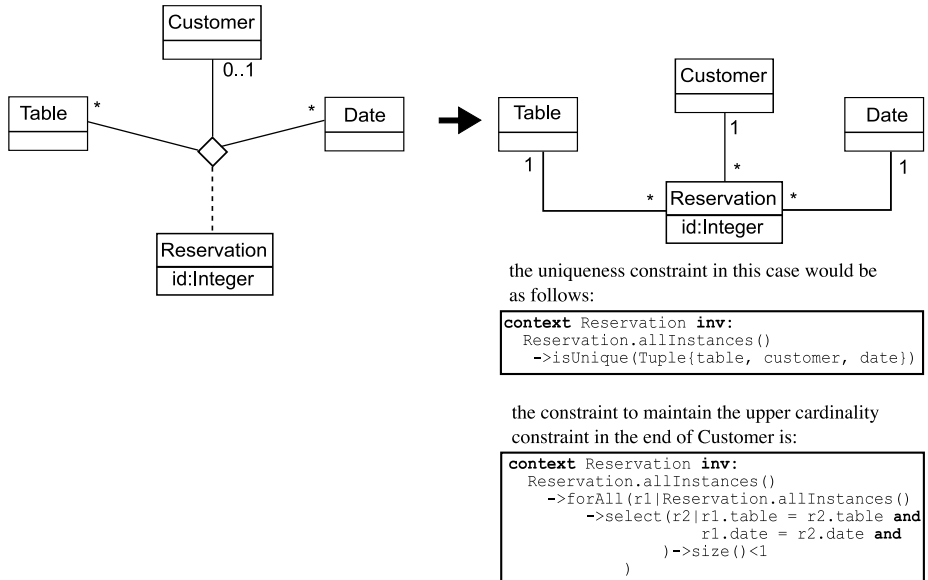


Figure 4.5: Implicit reification of a ternary relationship with association class.

Each of these relationships have the same cardinality constraints *1:Many* from `Table`, `Customer` and `Date` to `Reservation`. After having the new structure,

it's time to specify the uniqueness constraint which consists on the uniqueness of a tuple with three elements as shown in Fig. 4.5.

Finally, the upper cardinality constraint in the end of Customer have to be maintained. To do that, another constraint taking into account the semantics of such cardinality constraint is also shown in Fig. 4.5.

As the reader can observe, the implicit reification of association classes solves the problem of how to use the metrics with association classes. However, a new doubt appears about whether the relevance of the new entity types representing the reified association classes should be computed or not. After some evaluations, we decide to include such entities in the importance computation process because, as entity types, the concepts they represent really matter.

4.3.2 The Power of OCL

The schema rules previously defined are now taken into account. Here we present a new approach to include the knowledge provided by such rules.

One of the most important type of elements in conceptual schemas are relationship types between entity types. Such relationships are the key in methods based on link analysis. Our main goal here is to extract additional links between entity types from the schema rules.

As explained earlier in this document, the Object Constraint Language provides a powerful textual notation to specify schema rules in a declarative way. The notation of the OCL includes a set of expressions (see Fig 4.6), some of them are used to describe navigations between elements of the schema (e.g., access to attributes and navigation to association ends with *PropertyCallExp* expression of Fig. 4.7)

Furthermore, inside the body of a schema rule specified in OCL a set of entity types are referenced through association ends owned by them, access to attributes of other entity types, call to operations, and so on. Such set of references can be used in the relevance computation process.

Each schema rule is defined in the context of an entity type. If we go through the schema rules defined in previous sections and chapters we note that the first part of each rule is a *context* environment where to declare the entity that owns the rule. Of course, if we want to specify an invariant for to constraint the values of an attribute, it is clear that the context of such rule will be the entity type that owns the attribute.

Thus, if we mix the references with the concept of context we obtain a set of links joining the context entity type with each of the referenced (let's say participants) entity types in the body of the rule.

The new uncovered links act as virtual edges in the graph of entities and relationships types that conforms the conceptual schema. Such edges must be

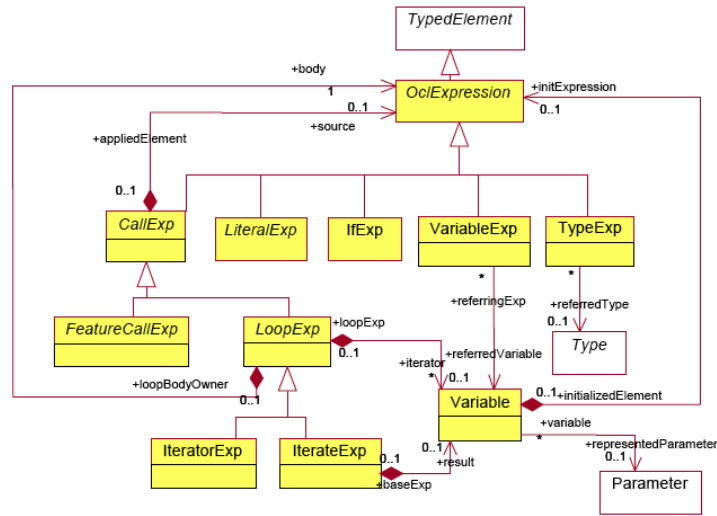


Figure 4.6: Fragment of the OCL metamodel including and overview of the expressions. Extracted from [38].

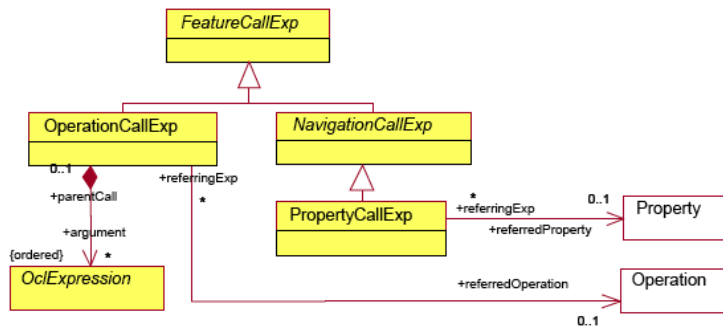


Figure 4.7: Fragment of the OCL metamodel including navigation expressions. Extracted from [38].

included in the computation process in order to take into account the knowledge added by invariants, derivation rules, and pre- and postconditions, all of them written in OCL.

To understand our approach we show an example in Fig. 4.8 with a simple schema containing three entity types and two relationships. The top of the figure describes a schema rule that check the number of seats of a room. Concretely, such number must be greater or equal than the number of assistants (the audience) of the related meeting. Furthermore, in the right side of the top of the figure there is a breakdown of the different sections of the OCL expressions to show the referenced entity types.

It is possible to distinguish between the entity types that participate in the

body of this schema rule (indicated with dashed lines) and the context entity type (indicated with a dotted line). Also, bold arrows indicate the structural navigations through relationship types in the schema rule (from self, i.e. Room, to Meeting, and from Meeting to Person using the rolename).

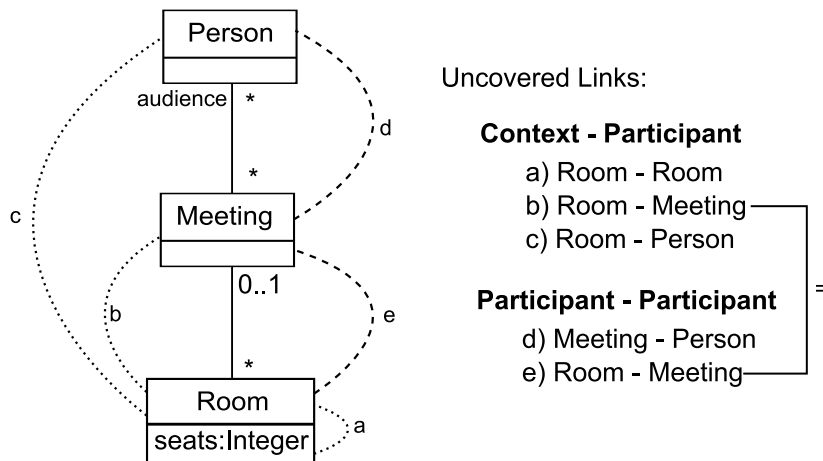
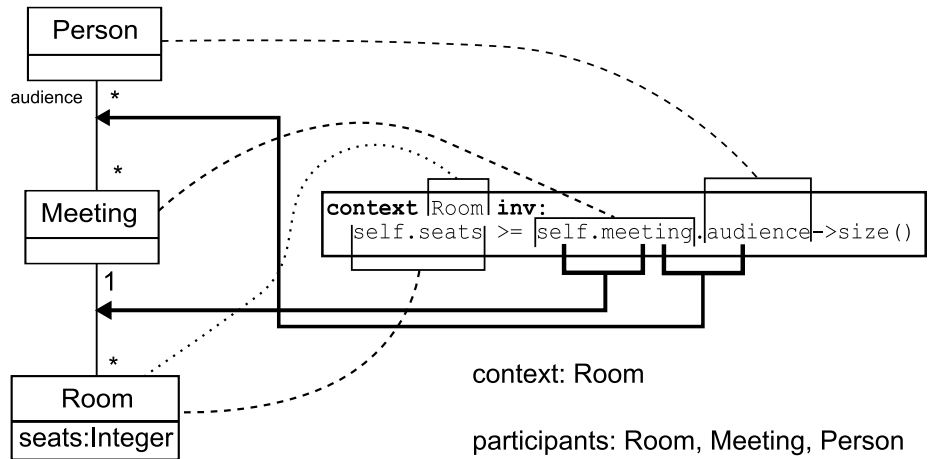


Figure 4.8: Example of uncovered links extracted from the OCL.

On one side, our approach consists on create new virtual links between the context entity type and each one of the participants. We name these kind of links *Context-Participant Non-Structural* links (CPNS). It is important to note that although an entity type may participate more than once in the same body of a schema rule, only one CPNS link is created between the context entity type an such participant entity type to avoid deviation. It is possible to see the CPNS links of the example in the right side of the bottom part of Fig. 4.8 (and with dotted lines in the schema).

On the other hand, we also create virtual links between the participants

that appear as source and target in a navigation expression of the OCL. In the example of Fig. 4.8, the bold arrows of the top indicate two structural navigations. We mean structural navigations to such navigations through relationship types. In this case we have the OCL expression `self.meeting`, which indicates a navigation through the relationship between Room and Meeting (`self` references the context of the schema rule, in this case, Room). Also we have `meeting.audience` that shows the navigation through the relationship between Meeting and Person.

Then, our approach consists on create new special links to note the use of navigations through relationships in the body of a schema rule. We name these kind of links *Participant-Participant Structural* links (PPS). It is possible to see them in the bottom part of Fig. 4.8, indicated with dashed lines.

Finally, once we have extracted the CPNS and PPS links from the schema rule, next step consists on avoid repetitions of links. Therefore, the final set of uncovered links from a schema rule is the union of both sets. This way, the repeated link Room-Meeting in the example will appear only once.

To sum up, the steps we have follow to obtain new links between entity types from a schema rule are as follows:

1. Detect the context entity type of the schema rule.
2. Detect the referenced entity types in the body of the schema rule.
3. Construct virtual links between the context and the referenced entity types (CPNS links).
4. Construct virtual links between the entity types that participate in navigation expressions of the OCL (PPS links).
5. Apply the union operation with the previous sets to delete repetition of links.
6. The result of the previous step is the set of uncovered links of the schema rule

Next, we will introduce new metrics that will take into account the extracted links from each one of the schema rules of the conceptual schema.

Another knowledge in the schema that has not been considered before is the information provided by cardinality constraints in association ends of relationship types. Such information can be added to the relevance computation process of entity types using the OCL.

Cardinality constraints can be converted into schema rules specified in OCL. The way to do it is very simple. It only consists on create an invariant whose context will be the entity type in the opposite side of the cardinality constraint, for each cardinality constraint with a lower multiplicity greater than zero. The

same has to be done with cardinality constraints with an upper multiplicity distinct than the asterisk (*).

As an example, if we go back to the example of Fig. 4.8, there is a cardinality constraint whose upper multiplicity is 1 (the one in the side of Meeting). Therefore, the new schema rule is as follows:

```
context Room inv: self.meeting->size()<=1
```

Thus, this schema rule will be processed according to the steps explained before and the obtained links will be added into the calculation of the values for the extended metrics of the next section.

4.3.3 Extended metrics

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 [42]. 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 occurrence. The effect is declaratively defined by the postcondition of the operation, which is specified in OCL (see chp. 11 of [40]). 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} (Schema Rules) the set of constraints, derivation rules and pre- and postconditions. Each rule $sr \in \mathcal{SR}$ is defined in the context of an entity type, denoted by $context(sr)$. In OCL, each rule sr consists of a set of OCL expressions (see OCL [38]) which we denote by $expr(sr)$. 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 sr is denoted by $ref(sr)$.

We also include in \mathcal{SR} the schema rules corresponding to the equivalent OCL invariants of the cardinality constraints. For example, in Fig. 4.9 the cardinality “1..*” between Company and Employee is transformed into the invariant:

```
context Company inv: self.employee->size()>=1
```

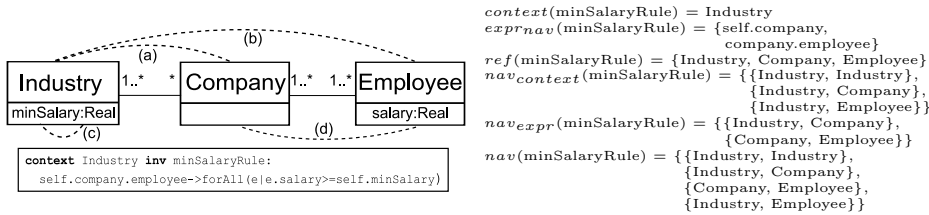
A special kind of OCL expression is the navigation expression that define a schema navigation from an entity type to another through an association (see *NavigationCallExp* of OCL in [38]). We use $expr_{nav}(sr)$ to indicate the navigation expressions inside a rule $sr \in \mathcal{SR}$. Such expressions only contain two entity types as its participants, i.e. the *source* entity type and the *target* one (see the example in Fig. 4.9).

4.3. Extended Measures

Notation	Definition
$context(sr)$	$= e \in \mathcal{E} \mid sr \in \mathcal{SR} \wedge sr \text{ DefinedIn } e$
$members(exp)$	$= \{e \in \mathcal{E} \mid e \text{ is a participant of } exp\}$
$expr(sr)$	$= \{expr \mid expr \text{ is contained in } sr\}$
$ref(sr)$	$= \cup_{exp \in expr(sr)} \{members(exp)\}$
$expr_{nav}(sr)$	$= \{expr \in expr(sr) \mid expr \text{ is a navigation expression}\}$
$nav_{expr}(sr)$	$= \cup_{exp \in expr_{nav}(sr)} \{\{e, e'\} \subset \mathcal{E} \mid \{e, e'\} = members(exp)\}$
$nav_{context}(sr)$	$= \{\{e, e'\} \subset \mathcal{E} \mid e = context(sr) \wedge e' \in ref(sr)\}$
$nav(sr)$	$= nav_{context}(sr) \cup nav_{expr}(sr)$
$rconn(e)$	$= \uplus_{sr \in \mathcal{SR}} \{e' \in \mathcal{E} \mid \{e, e'\} \subset nav(sr)\}^2$
$rconn_{inh}(e)$	$= rconn(e) \uplus \{rconn_{inh}(e') \mid e' \in par(e)\}$

Table 4.9: Definition of extended metrics.

We denote by $nav_{expr}(sr)$ the set of pairs that participate in the navigation expressions of $sr \in \mathcal{SR}$. We also denote by $nav_{context}(sr)$ the sets of pairs of entity types composed by the context of the rule sr and every one of the participant entity types of such rule ($e \in ref(sr)$). Finally, we define $nav(sr)$ as the union of $nav_{context}(sr)$ with $nav_{expr}(sr)$ and, $rconn(e)$ as the multiset of entity types that compose a pair with e in $nav(sr)$. Note that since we use \uplus , $rconn(e)$ may contain duplicates because it takes into account each rule sr 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.



²Note that “ \uplus ” denotes the multiset (or bag) union that produces a multiset as in $\{a, b\} \uplus \{a\} = \{a, a, b\}$. Apart from it, we also force that $rconn(e) = \emptyset$ (empty set) if $conn_{inh}(e) = \emptyset$.

4.4 Extended Methods

This section presents the extended versions of the methods introduced in Section 4.2. Such extensions take into account the schema rules, including the conversion of cardinality constraints into OCL invariants.

We formally define the new components of each method and use an extended schema with a set of schema rules to increase the understandability of the methods. Such example is a reformulation of the one in Fig. 4.2, shown at Fig. 4.10.

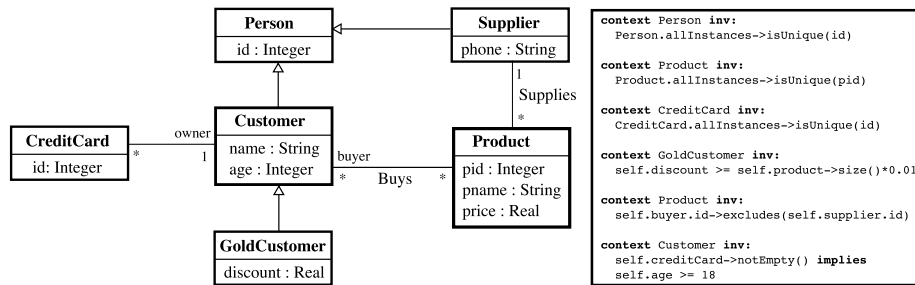


Figure 4.10: Extension of the schema in Fig.4.2 with some OCL invariants.

The conversion of the cardinality constraints of the schema in Fig. 4.10 is as follows:

```

context Product inv: self.supplier->size()>=1
context Product inv: self.supplier->size()<=1
context CreditCard inv: self.employee->size()>=1
context CreditCard inv: self.customer->size()<=1
  
```

It is important to note that we create a new schema rule defined in OCL for each cardinality constraint. A possible change here could be were the lower and upper values of a multiplicity have the same value, as in the previous example where the value is 1. Thus, the conversion will result in:

```

context Product inv: self.supplier->size()==1
context CreditCard inv: self.employee->size()==1
  
```

However, we select to convert as in the first case because if we would have a lower value greater than zero and an upper value distinct than asterisk, and such values were different, then the conversion in this case will produce a pair of schema rules, while if the values of the upper and the lower values are equal it produces only one schema rule. To be consistent, we prefer to create a schema rule for each upper or lower value.

4.4.1 The Connectivity Counter Extended (CC+)

Our extension to this method follows the same idea than the base version but also including the number of participations of each entity type in the navigation relationships extracted from 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 (in case of such events were defined). Formally:

$$I_{CC}^+(e) = |assoc(e)| + |rconn(e)|$$

Table 4.10 presents the obtained results once this method has been applied to example of Fig 4.10. One more time, we conclude that Product and Customer are the most relevant entity types.

e	$ assoc(e) $	$ rconn(e) $	$I_{CC}^+(e)$
CreditCard	1	6	7
Customer	2	10	12
GoldCustomer	0	4	4
Person	0	0	0
Product	2	14	16
Supplier	1	4	5

Table 4.10: Results for CC+ applied to example of Fig 4.10.

Note that although the value of $|rconn(Person)|$ is not zero, there is a condition into the extended metrics to be consistent. Such condition states that the entity types that are not directly or indirectly (with inherited associations from their parents) connected through relationships with other entities ($conn_{inh}(e) = \emptyset$), must have a $rconn$ equal to the empty set. The cause of it is to avoid give importance to entity types from virtual connections (that should enforce real connections) when such entities are really unconnected.

4.4.2 The Simple Method Extended (SM+)

As in the extended version of the Connectivity Counter, the Simple Method has been extended including the number of participations of each entity type in the navigation relationships extracted from the schema rules specification (and cardinality constraints). We also take into account (in $|assoc(e)|$) the associations of each entity type e with the event types of the behavioural schema (in case of such events were defined). Formally:

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

For example, if this extended version of the simple method, and also the simple method, were applied into the schema shown in Fig. 4.9, we would have

$I_{SM}(\text{Company})=2$ and $I_{SM}^+(\text{Company})=8$, because $|par(\text{Company})|=|chi(\text{Company})|=|attr(\text{Company})|=0$, $|assoc(\text{Company})|=2$, and $|rconn(\text{Company})|=6$, 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.

The values obtained after the application of the extended version of the simple method to the previous schema shown at Fig 4.10 are indicated in Table 4.11. There, it is possible to check that the most important entity type is Customer followed by Product.

e	$ par(e) $	$ chi(e) $	$ attr(e) $	$ assoc(e) $	$ rconn(e) $	$I_{SM}^+(e)$
CreditCard	0	0	1	1	6	8
Customer	1	1	2	2	10	16
GoldCustomer	1	0	1	0	4	6
Person	0	2	1	0	0	3
Product	0	0	3	2	14	19
Supplier	1	0	1	1	4	7

Table 4.11: Results for SM+ applied to example of Fig 4.10.

As the reader can observe, it is possible to say that the Simple Method is also an extended version of the Connectivity Counter (in both base and extended cases).

4.4.3 The Weighted Simple Method Extended (WSM+)

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:

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

We use the same strengths than in the base version of the method. Concretely $q_{attr} = 1$, $q_{inh} = 0.6$, and $q_{assoc} = 0.4$. Furthermore, we use for q_{rule} the same strength as q_{assoc} . Although it is possible to use a different strength for q_{rule} , since the behaviour of $|rconn(e)|$ is similar than for $|assoc(e)|$ due to both represent participations of entities, we decided to use the same strength.

The results of the application of the weighted simple method to the example in Fig 4.10 are shown in Table 4.12. The most relevant entity type is already Product, followed by Customer. As the reader can observe, this pair of entities are selected by all the methods presented here as the most important ones. The littleness of the example is the main cause of this behaviour.

e	$q_{inh}(par(e) + chi(e))$	$q_{attr} attr(e) $	$q_{assoc} assoc(e) $
CreditCard	0.6x0	1x1	0.4x1
Customer	0.6x2	1x2	0.4x2
GoldCustomer	0.6x1	1x1	0.4x0
Person	0.6x2	1x1	0.4x0
Product	0.6x0	1x3	0.4x2
Supplier	0.6x1	1x1	0.4x

e	$q_{rule} rconn(e) $	$I_{WSM}^+(e)$
CreditCard	0.4x6	3.8
Customer	0.4x10	8
GoldCustomer	0.4x4	3.2
Person	0.4x0	2.2
Product	0.4x14	9.4
Supplier	0.4x4	3.6

Table 4.12: Results for WSM+ applied to example of Fig 4.10.

4.4.4 The Transitive inheritance Method Extended (TIM+)

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:

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

As before, the results of the application of the TIM+ method to the previous example are shown in Table 4.13. Here, the most important entity type is GoldCustomer, followed by Product and Customer.

4.4.5 EntityRank Extended (ER+)

In our extension to the EntityRank method 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:

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

If we fix that $\sum_{e \in \mathcal{E}} I_{ER}^+(e) = 1$ then the results obtained are shown in Table 4.14. Remember that we have chosen a $q = 0.15$, which is a common value in the literature.

e	$ par_{inh}(e) $	$ chi_{inh}(e) $	$ attr_{inh}(e) $
CreditCard	0	0	1
Customer	1	1	3
GoldCustomer	2	0	4
Person	0	3	1
Product	0	0	3
Supplier	1	0	2

e	$ assoc_{inh}(e) $	$ rconn_{inh}(e) $	$I_{TIM}^+(e)$
CreditCard	1	6	8
Customer	2	10	17
GoldCustomer	2	14	22
Person	0	0	4
Product	2	14	19
Supplier	1	4	8

Table 4.13: Results for TIM+ applied to example of Fig 4.10.

e	$I_{ER}^+(e)$
CreditCard	0.15
Customer	0.25
GoldCustomer	0.11
Person	0.03
Product	0.34
Supplier	0.12

Table 4.14: Results for ER+ applied to example of Fig 4.10.

4.4.6 BEntityRank Extended (BER+)

Our extension for the BEntityRank method is in the same way as in ER+. The difference between the formula of ER+ and BER+ is that BER+ takes into account the values of the attribute measure of BEntityRank. The definition is:

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

Similarly than other methods based on link analysis, to compute the importance of the entity types we need to solve an equation system (the same as in BER but including the new components provided by $rconn$ measure). We fix that $\sum_{e \in \mathcal{E}} I_{BER}^+(e) = 1$ then the results obtained are shown in Table 4.15. We have already chosen a $q = 0.15$, which is a common value in the literature.

e	$I_{BER}^+(e)$
CreditCard	0.15
Customer	0.26
GoldCustomer	0.10
Person	0.02
Product	0.36
Supplier	0.11

Table 4.15: Results for BER+ applied to example of Fig 4.10.

4.4.7 CEntityRank Extended (CER+)

One more time, our extension includes the uncovered navigations of the schema rules as bidirectional transitions for the random surfer. The new definition is the same as for CER but including the measure of the knowledge provided by the schema rules:

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

We already fix that $\sum_{e \in \mathcal{E}} I_{CER}^+(e) = 1$ then the results obtained are shown in Table 4.16. We have chosen $q_1 = 0.1$ and $q_2 = 0.2$, which are some good values as indicated in [55].

As in the previous methods the most important pair of entity types are Product and Customer, and we have obtained similar rankings because the schema shown in Fig 4.10 is very small. The methods presented here must be tested with large schemas to bring out the differences between them.

e	$I_{CER}^+(e)$
CreditCard	0.12
Customer	0.27
GoldCustomer	0.12
Person	0.06
Product	0.31
Supplier	0.12

Table 4.16: Results for CER+ applied to example of Fig 4.10.

4.5 Comparison

In the previous sections the algorithms that conform a sample of the existing methods in the literature about how to compute the importance of conceptual schema's entity types have been explained. Here we want to discover some differences between them.

After the study of each method and measure, it is possible to detect two main characteristics. On one hand, there are two types of methods –those based on occurrence counting and those that follow the link analysis approach. Take a look at Table 4.17 to see this classification.

OCCURRENCE COUNTING	LINK ANALYSIS
CC	ER
SM	BER
WSM	CER
TIM	
CC+	ER+
SM+	BER+
WSM+	CER+
TIM+	

Table 4.17: Classification of selected methods according to their approach.

Methods based on occurrence counting are similar to those methods in the field of text searching that count the frequency of words to select the most important ones. Remember the example shown in Fig. 2.8 of Chpt. 2. The difference here is that these methods count for each entity type the number of related elements to it (or owned by it, as in the case of attributes), plus the number of occurrences of such entity type in schema rules and in the conversion of cardinality constraints to rules.

This way, we follow the principle of high appearance –the more occurrences an item has in an scope, the more important such item becomes.

Methods based on link analysis take into account the importance of the other related elements to the current one because the importance of the current is an addition of fragments of the importance of the others. Remember the example with smileys in Fig 3.10 of Chpt. 3. Thus, methods based on link analysis approach need special iterative algorithms to solve the importance computation based on a system of equations.

Such iterative methods attempt to solve a problem by finding successive approximations to the solution starting from an initial point. These iterative solvers, like the Jacobi algorithm, the power method or the inverse power method (see [56]), are computationally expensive and slower than the methods

based on occurrence counting. However, solutions obtained from these methods have better results taking into account lower amounts of knowledge, as we will see in next chapter.

On the other hand, there exists another difference between base methods and extended ones: the amount of knowledge taken into account in the computation process of the relevance of entity types.

As explained in the previous chapters, methods to calculate the relevant entity types of a conceptual schema use some metrics whose values are gathered from the structural subschema. Our approach introduces some new measures to convert the knowledge within the whole conceptual schema (including the behavioural subschema) into meters for relevance.

Table 4.18 shows the different elements of knowledge of the conceptual schema taken into account for each of the (base and extended) methods. Cells with an x indicate that the algorithm in the row uses the information of the column.

First four columns include entity and relationship types, generalization/specialization relationships and attributes in the structural schema. In the base versions, the methods only use the knowledge of such columns depicted in the structural schema. Extended versions also use the entity and relationship types, attributes and generalizations included in the behavioural schema, and therefore the schema rules and elements of the complete conceptual schema.

Last columns of Table 4.18 represent the knowledge extracted from the schema rules, including derivation rules, invariants, pre- and postconditions and, of course, the conversion of cardinality constraints into schema rules.

First rows in the table shows the base versions of the methods. Such methods only use information from the structural schema. As the event types are part of the behavioural schema, they are not applicable in this part of the table. Finally, last rows are the extended versions of the algorithms in the first rows.

The gap in the rows of *CC*, *ER* and *BER* methods referencing the use of generalization relationships and attributes is maintained in the extended versions *CC+*, *ER+* and *BER+* to follow the same approach than in the base versions. In the case of *ER+* and *BER+*, the complete link analysis-based method that takes into account the whole knowledge is the extension of the CEntityRank (*CER+*).

It is really clear that the extended versions of the selected methods from the literature take into account more knowledge from the conceptual schema than the base versions. Therefore, according to the *Principle of High Appearance in Conceptual Schemas*, the results obtained with the extended versions are more realistic because there is more amount of knowledge that contributes to calculate the importance of entity types.

KNOWLEDGE									
BASE METHODS (Structural Schema)	Entity Types	Relationship Types	Generalization Relationships	Attributes	Cardinality Constraints	Derivation Rules	Invariants	Pre- and Postconditions	Event Types
	CC	x	x						
SM	x	x	x	x					n/a
WSM	x	x	x	x					n/a
TIM	x	x	x	x					n/a
ER	x	x							n/a
BER	x	x		x					n/a
CER	x	x	x	x					n/a
EXTENDED METHODS (Complete Schema)									
CC+	x	x			x	x	x	x	x
SM+	x	x	x	x	x	x	x	x	x
WSM+	x	x	x	x	x	x	x	x	x
TIM+	x	x	x	x	x	x	x	x	x
ER+	x	x			x	x	x	x	x
BER+	x	x		x	x	x	x	x	x
CER+	x	x	x	x	x	x	x	x	x

Table 4.18: Comparison of knowledge used between both base and extended versions of the selected methods.

Chapter 5

Experimental Evaluation

We have implemented the 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 [52] and the UML metaschema [8, 9]. 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. On the other hand, to evaluate the extended versions we have use the complete conceptual 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). For the UML metaschema there is no behavioral schema and therefore we have only used the complete structural schema (also taking into account schema rules and cardinality constraints).

Section 5.1 introduces the two conceptual schemas selected for the evaluation of the methods explained in the previous chapter. Section 5.2 summarizes the main conclusions we have drawn from the study of the correlation between the results of each of the methods. Section 5.4 studies the differences of execution time for the previous methods in order to extract some ideas about in which situations could be preferably to use one method or another. And finally Section 5.4 presents a comparison between the most important entity types obtained from the methods.

5.1 Conceptual Schemas for the Evaluation

Large conceptual schemas are more common than it seems at first. There exists an important number of organizations that work with big schemas but to get access to them is not an easy task due to such organizations consider their schemas as one of the most valuable artifacts they have and, therefore, their

5.1. Conceptual Schemas for the Evaluation

aim is to protect the schemas from external persons.

Fortunately, the research group I belong to (GMC¹) has a conceptual schema to make tests with it for the research of the group –the conceptual schema of the osCommerce [52].

Furthermore, we have found another conceptual schema that should be useful to check the behaviour of the methods to compute the importance of entity types. It is an implementation of the UML metaschema [39], which can be considered as a large schema because of it contains a big amount of entity types, although the most well-known part was the *kernel* package, containing concepts like class, association and so on.

5.1.1 The osCommerce

The osCommerce is an open source solution for e-commerce portals that can be adapted to the requirements of every business and each geographical zone. It is composed by two sections: an e-commerce portal or online store, and the administration of it.

The online store allow users to check the available products of the seller and purchase them. On the other hand, the administration section provides a flexible and adaptable way of configure the portal according to the needs of the owner of the store, including customization of taxes to apply, languages to support, *look&feel* of the portal and so on. Look at Fig. 5.1 to see an example of the osCommerce page for to confirm an order.

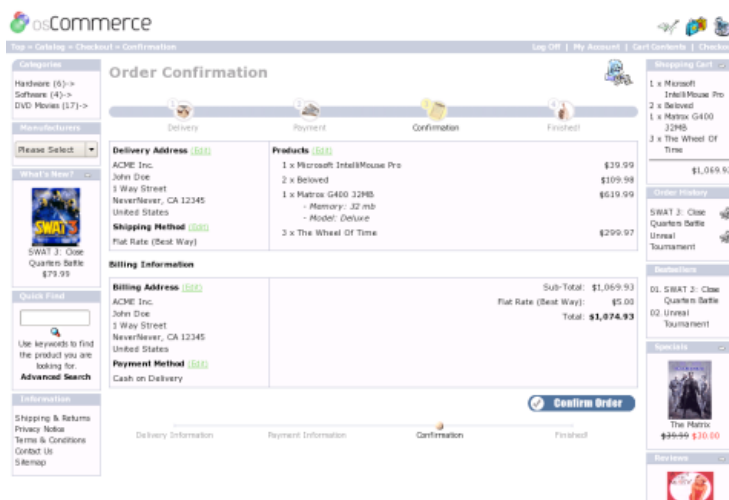


Figure 5.1: Order confirmation at osCommerce.

¹<http://guifre.lsi.upc.edu/index.html>

The osCommerce is maintained by the osCommerce Team and the information of the project can be found on the web². The osCommerce was started in March 2000 and has since matured to a solution that is currently powering 14,112 registered live shops around the world.

The conceptual schema of the osCommerce was the result of the final project of Tort for his degree in informatics [52]. It comprises 346 entity types (of which 261 are event types belonging to the behavioural schema), 458 attributes, 183 associations, 204 general constraints and derivation rules and 220 pre- and post conditions.

The structural schema of the osCommerce contains the concepts that are useful to maintain in the field of the e-commerce. Such concepts, represented as entity types and the relationships between them are shown in a reduced way at Fig. 5.2. The entities that appear there are supposed to be the most relevant ones, although this diagram was manually constructed by Tort and Olivé in [52] without the aid of any of our methods.

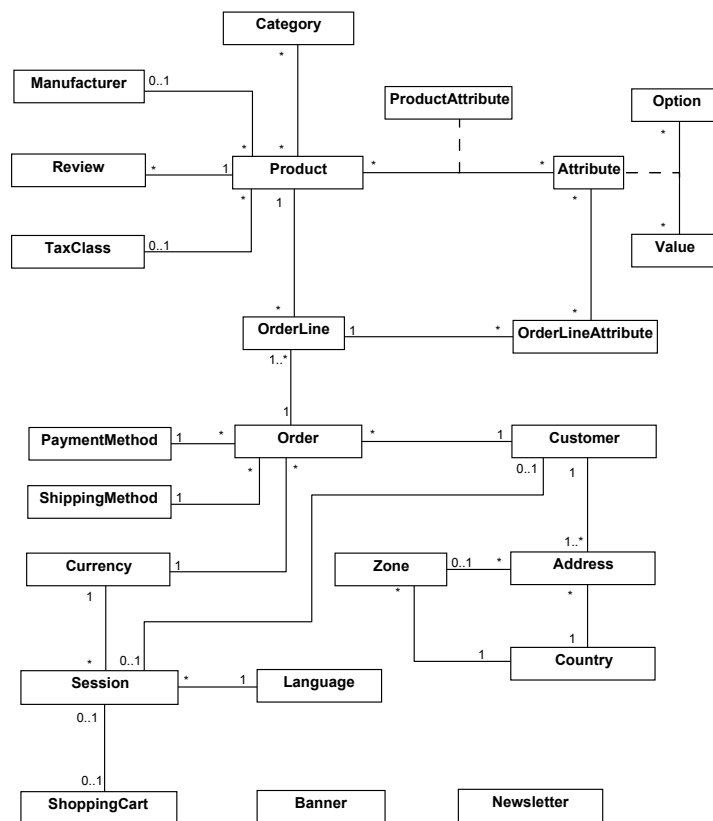


Figure 5.2: Simplification of the structural schema of the osCommerce. Extracted from [52]

²<http://www.oscommerce.com/>

The main characteristic of the conceptual schema of the osCommerce is that the event types of the behavioural schema are available and, therefore, the knowledge they provide can be included in the importance computation process. Such schema will be totally processed by the extended versions of the methods of the previous chapter. Thus, we consider the osCommerce as a complete and large conceptual schema.

Each event type of the behavioural schema has its own structure, integrity constraints and effect defined in UML/OCL and then our methods are able to work with them. Look at the example of event type at Fig. 5.3.

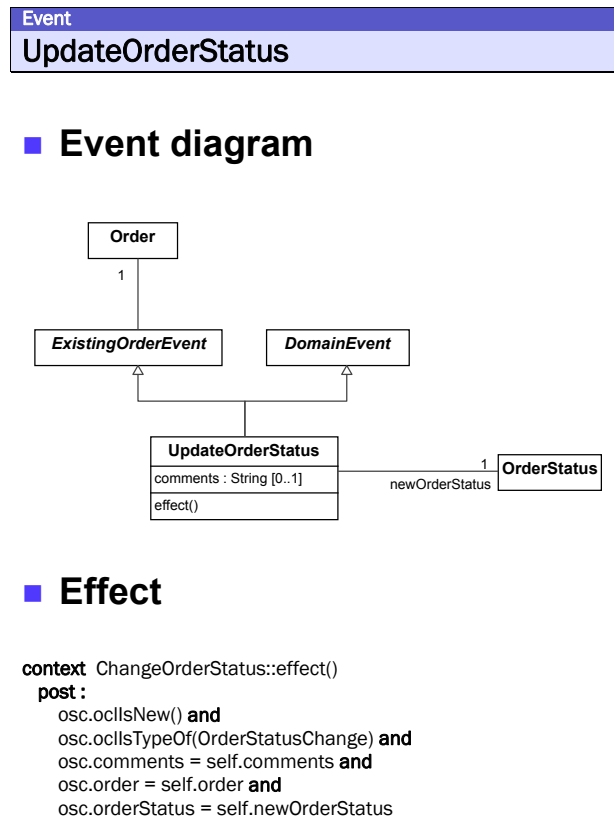


Figure 5.3: UpdateOrderStatus event at osCommerce. Extracted from [52]

Since the event types are modeled as entity types, the same methods to compute the importance of entity types can be applied to compute the importance of event types. Furthermore, we can obtain two rankings –one for the most relevant entity types and other for the most relevant event types.

5.1.2 The UML Metaschema

On the other hand, we have another large conceptual schema to proof the methods –the conceptual schema of the UML metaschema [39]. This time, such schema does not contain any information about event types. However its structural schema contains a big amount of entity types, relationships and schema rules.

The version of the UML metaschema we have used (see [8, 9]) comprises 293 entity types, 93 attributes, 377 associations, 54 derivation rules and 116 general constraints. A view of the kernel section of the UML metaschema can be found in Fig. 5.4. The kernel section is the most well-known section of this schema, although it worths mentioning that it is only a little fragment of the whole.

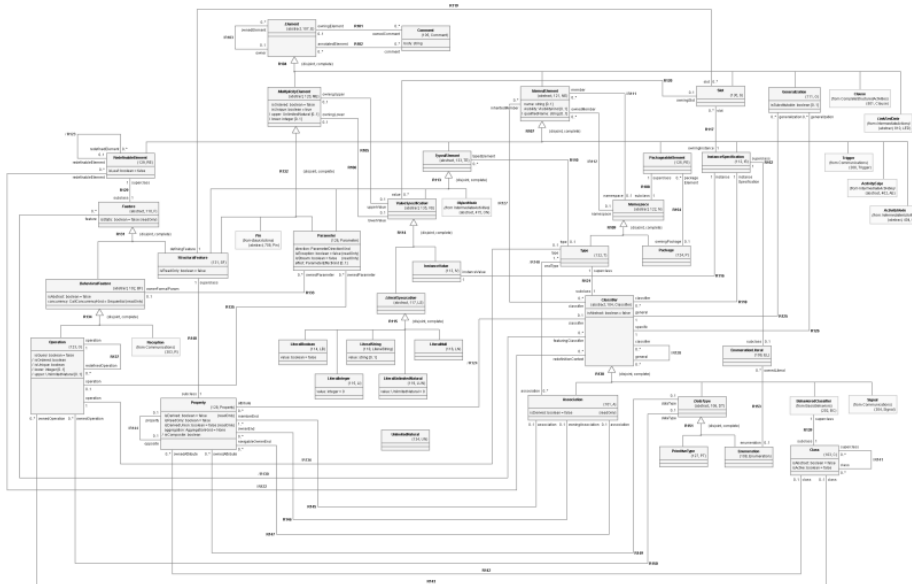


Figure 5.4: Shape of the Kernel section of the UML metaschema.

We define the connectivity degree δ of a conceptual schema as the factor that results from the fraction of the number of relationship types and the number of entity types. Formally:

$$\delta = \frac{|\mathcal{R}|}{|\mathcal{E}|}$$

Therefore, according to the information of our two schemas, the connectivity degree is shown at Table 5.1. It is easy to see that although the conceptual schema of the osCommerce includes the event types in the behavioural schema, the conceptual schema of the UML metaschema has a higher connectivity degree δ . Thus, the conceptual schema of the UML metaschema is a more connected schema.

Conceptual Schema	δ
osCommerce	$\frac{183}{346} \simeq 0.52$
UML Metaschema	$\frac{377}{293} \simeq 1,29$

Table 5.1: Connectivity degree δ for the test schemas.

5.2 Correlation Study

This section analyses the similarity between the different methods proposed to compute the importance of entity types. The main purpose here is to determine the behaviour of the methods in order to extract some useful conclusions.

Roughly, we define *correlation* as a measure that indicates the strength and direction of a linear relationship between two variables. The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship, and some value in between in all other cases, indicating the degree of linear dependence between the variables. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables. If the variables are independent then the correlation is 0.

This way, we compute the correlation between the results for each pair of methods in order to estimate whether the obtained rankings of entity types according to the computed relevance are dependent or not. If such results have a high degree of linear dependency, it is possible to say that the results are similar in both methods.

5.2.1 Correlation Between the Original and the Extended Versions

Firstly, we study the correlation between the original and the extended version for the methods described in the previous chapter. Our research wants to answer if there are some of the base versions that maintain a higher degree of similarity with their extended versions.

Figure 5.5 shows, for each method, the results obtained in the original and the extended versions for the conceptual schema of 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 importance of all entity types in each method is 100. Therefore we can consider the vertical axis as the percentage of relevance each entity type has.

It worths mentioning that for the case of the conceptual schema of the osCommerce, to compare the results of the base and extended versions we only use the structural schema for the base versions and the whole schema (containing

the behavioural schema, with event types) for the extended versions. However, in the comparison shown at Fig. 5.5 we only use 85 entity types that are the ones that belong to the structural schema due to the event types cannot be compared because the base versions only take into account the structural schema.

As shown in Fig. 5.5(g) 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 ($r=0.61$). 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 0.96 (see those results at Fig. 5.6).

An important behaviour is for the Connectivity Counter in the case of the conceptual schema of the osCommerce. The reader can observe at Fig. 5.5(a) that the correlation between the results obtained by CC and CC+ have a higher degree of linear dependency. This behaviour is a little bit lower in the case of the UML metaschema. It is possible to say that in the case of osCommerce, the schema has a lower degree of connectivity (see Table 5.1) therefore, the power given by relationships generates a higher amount of relevancy to those entity types that have a high number of participations in relationships. In the case of the UML metaschema, due to its higher connectivity degree, the importance of being connected is relatively lower because almost all the entities have participations in relationships.

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.

Obviously, the methods based on link analysis are more constant than those based on occurrence counting. The main reason for this behaviors is the recursive definition of its formulas. Such methods need the importance of other entity types in order to compute the importance of an entity. Therefore, it implies the existence of a dependency that implies an iterative computation from an initial point to achieve the convergence to the final result. In the case of occurrence counting methods, the computation of the relevance for an entity is totally independent from the other entity types.

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 [55], which, based on subjective evaluations given by expert evaluators (*oracles*), concludes that the method that gives the best results is the Simple Method. However, Fig. 5.5(b) shows that the result given by that method has a lower similarity when the whole schema knowledge is taken into account.

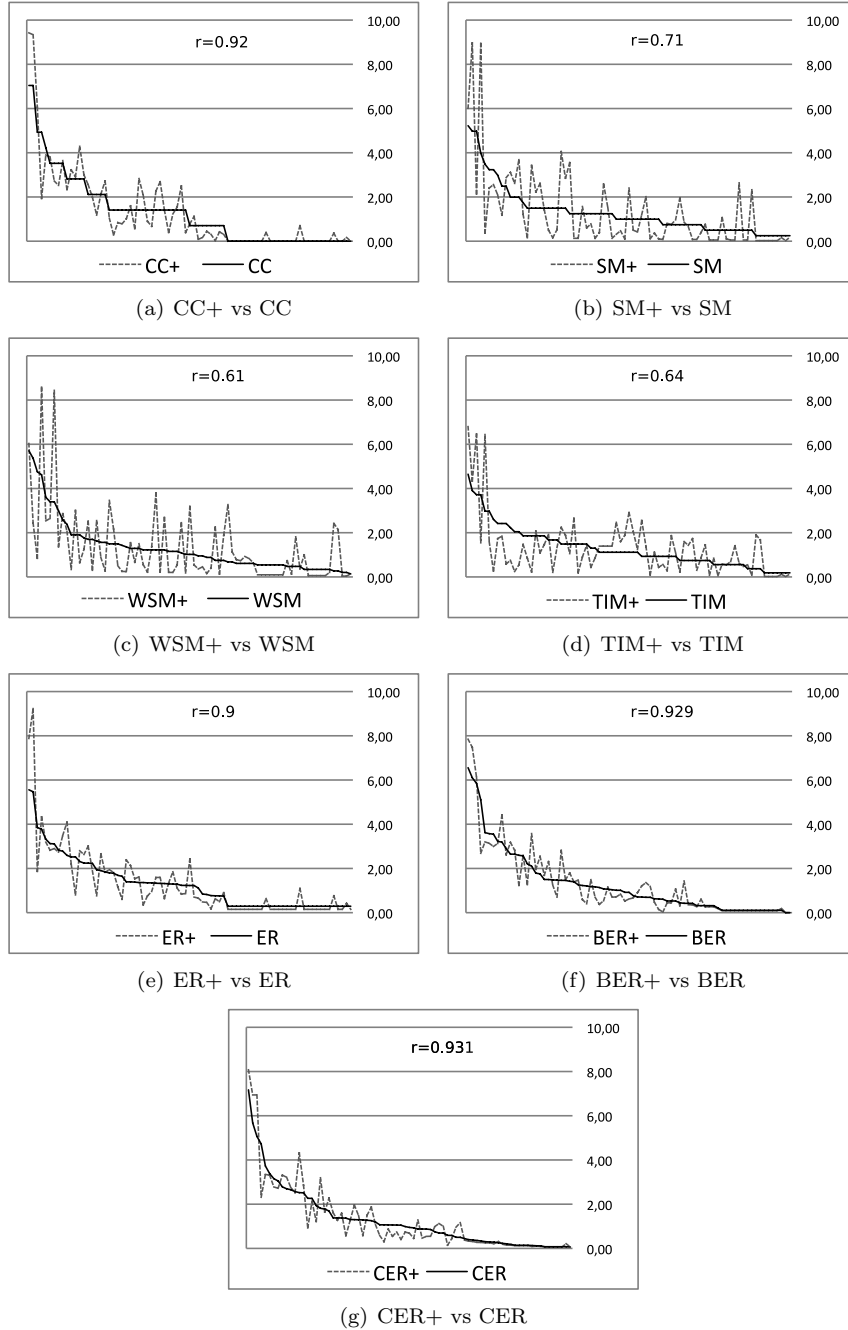


Figure 5.5: Comparison between base and extended importance-computing methods once applied to the osCommerce schema.

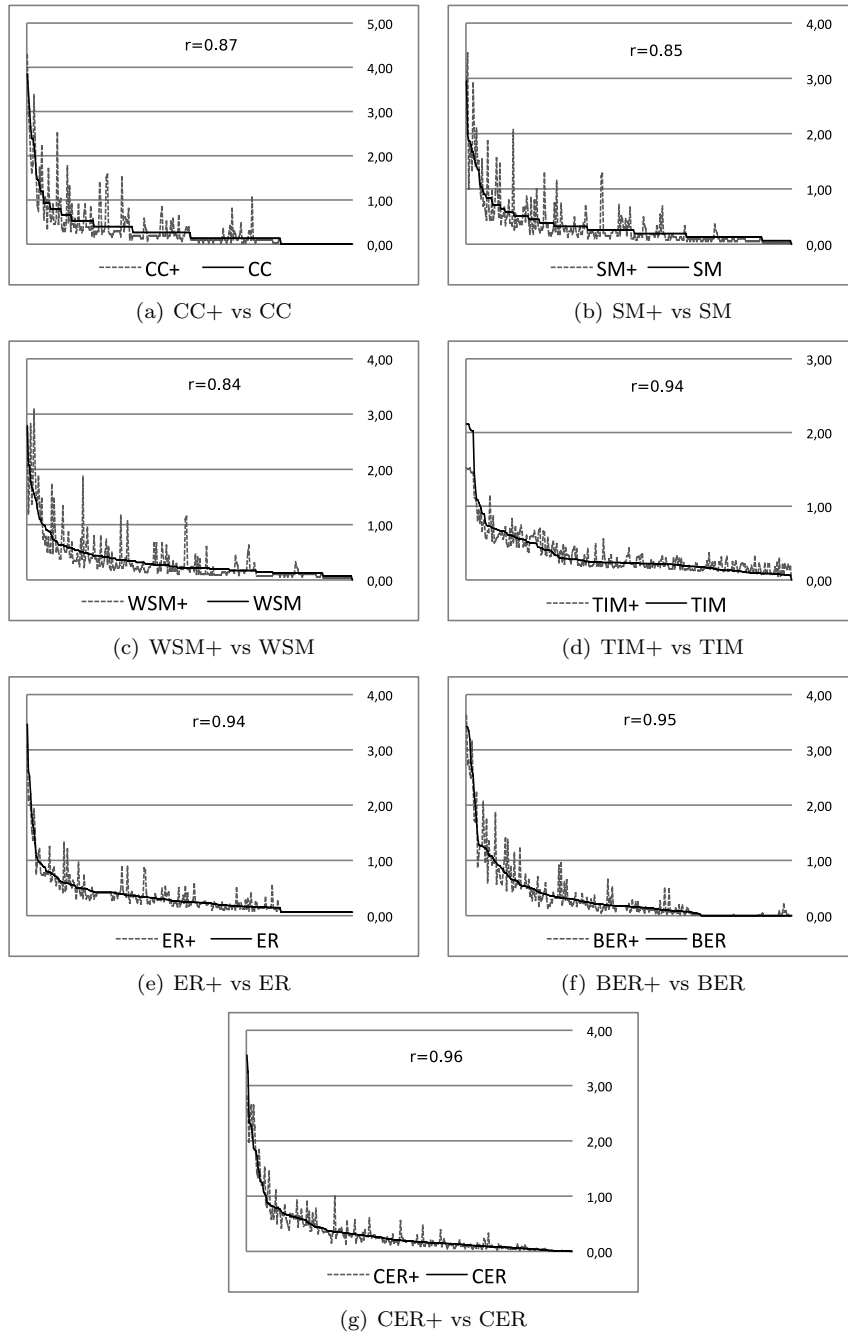


Figure 5.6: Comparison between base and extended importance-computing methods once applied to the UML metaschema.

5.2.2 Variability of the Original and the Extended Versions

Another interesting experiment is to study the correlation between base versions and extended versions separately. This way it is possible to compare the results of such methods and to search for a common behaviour according to if the methods take into account the complete conceptual schema or only the structural subschema.

Table 5.2 shows the correlation between each pair of methods (separately, originals and extended versions), in the case study of the UML metaschema. 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 0.73 to 0.98, while in the extended versions the range is from 0.81 to 0.99.

	I_{CC}	I_{SM}	I_{WSM}	I_{TIM}	I_{ER}	I_{BER}	I_{CER}
I_{CC}		0.87	0.79	0.06	0.96	0.85	0.86
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_{CC}^+	I_{SM}^+	I_{WSM}^+	I_{TIM}^+	I_{ER}^+	I_{BER}^+	I_{CER}^+
I_{CC}^+		0.99	0.97	0.23	0.93	0.82	0.81
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

Table 5.2: Correlation coefficients between results of original and extended methods for the UML metaschema.

On the other hand, Table 5.3 shows the correlation between each pair of methods (separately, originals and extended versions), in the case study of the conceptual schema of the osCommerce. 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 0.92 to 0.99. Furthermore, the correlation of all the pairs in which TIM appears have an increased value in the extended version.

Both tables (Table 5.2 and Table 5.3) are summarizations for the linear

regression plots contained inside Fig. 5.7 and Fig. 5.8, for the case of the UML metaschema, and Fig. 5.9 and Fig. 5.10, for the case of the conceptual schema of the osCommerce. It is important to note that in the linear regression plots we show the square of the correlation instead of the simple correlation of the previous tables.

	I_{CC}	I_{SM}	I_{WSM}	I_{TIM}	I_{ER}	I_{BER}	I_{CER}
I_{CC}		0.76	0.61	0.43	0.98	0.94	0.94
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_{CC}^+	I_{SM}^+	I_{WSM}^+	I_{TIM}^+	I_{ER}^+	I_{BER}^+	I_{CER}^+
I_{CC}^+		0.99	0.99	0.78	0.98	0.92	0.92
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

Table 5.3: Correlation coefficients between results of original and extended methods for the osCommerce.

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. We mean that the results obtained by extended versions have a lower degree of variance.

This conclusion is also significant because it assures that the use of the Simple Method (extended version) or also the Connectivity Counter extended (CC+) whose computational cost is very low, and on the other hand they allow the incremental recalculation of the importance of entity types when the schema changes, produces “good-enough” results.

Furthermore, we can conclude that the more knowledge of a conceptual schema the methods take into account to compute the importance of entity types, the more similar the obtained results will be. It induces an inverse proportional relationship between the quantity of knowledge and the variance of the results. This way, to choose a method or another we must study the amount of available knowledge we have for a conceptual schema.

5.2. Correlation Study

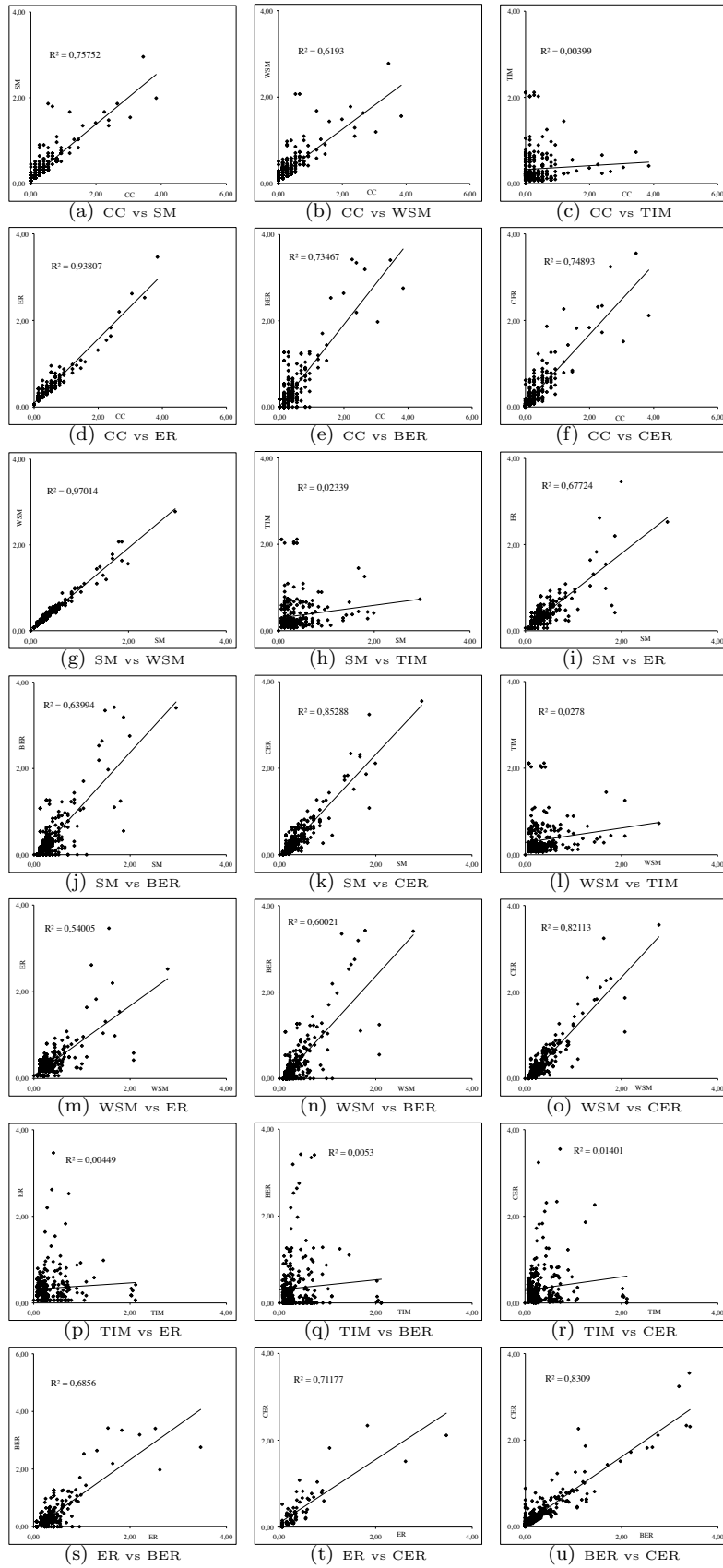


Figure 5.7: Comparison between base methods applied to the UML meta-schema.

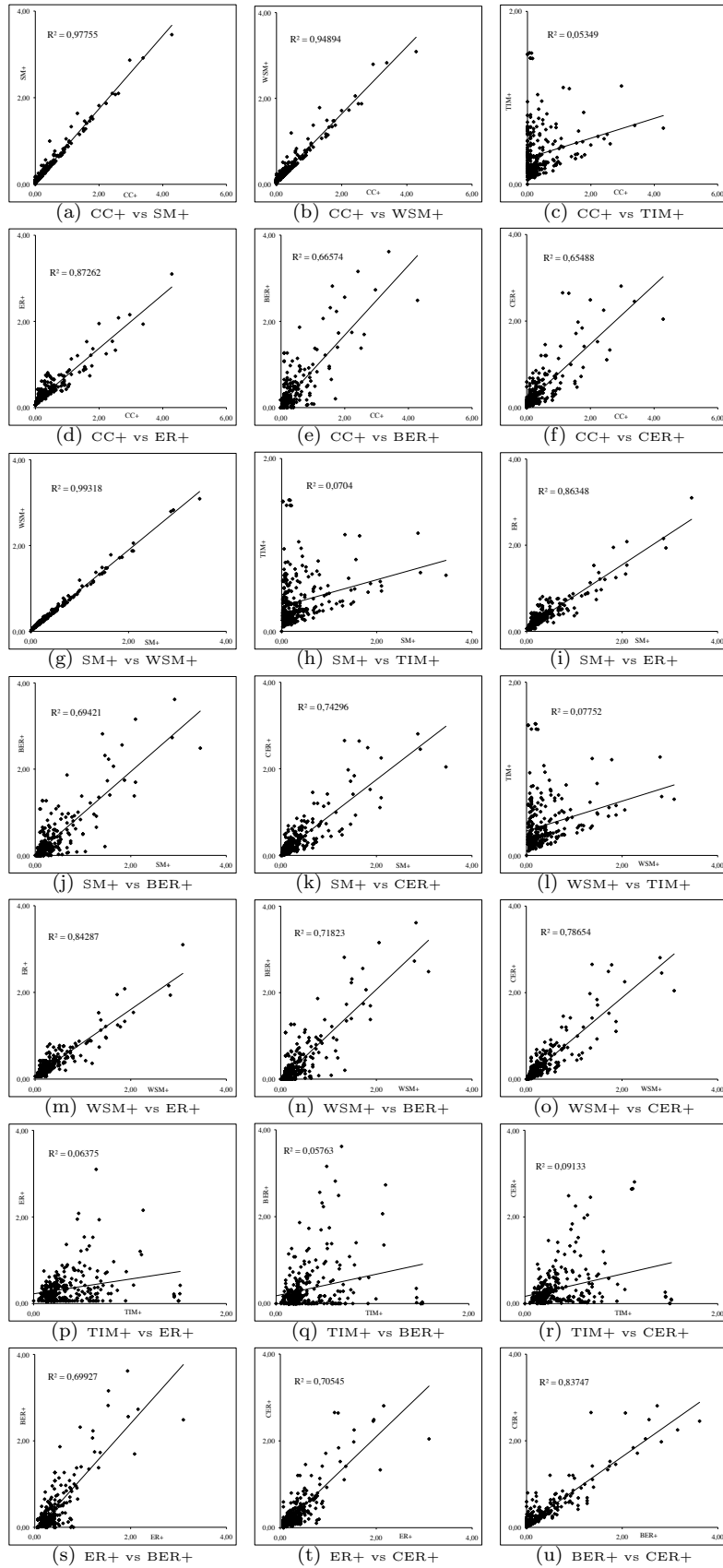


Figure 5.8: Comparison between extended methods applied to the UML meta-schema.

5.2. Correlation Study

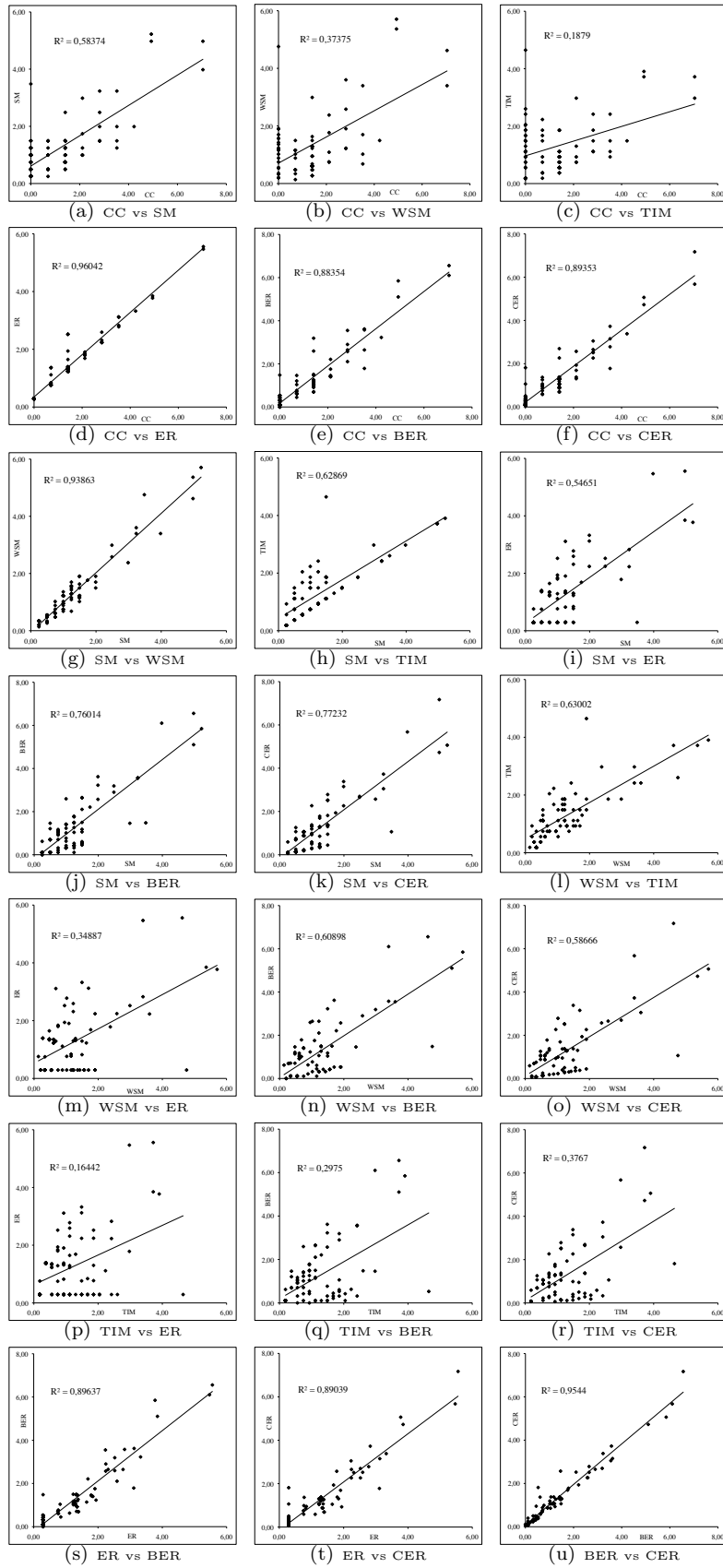


Figure 5.9: Comparison between base methods applied to the osCommerce.

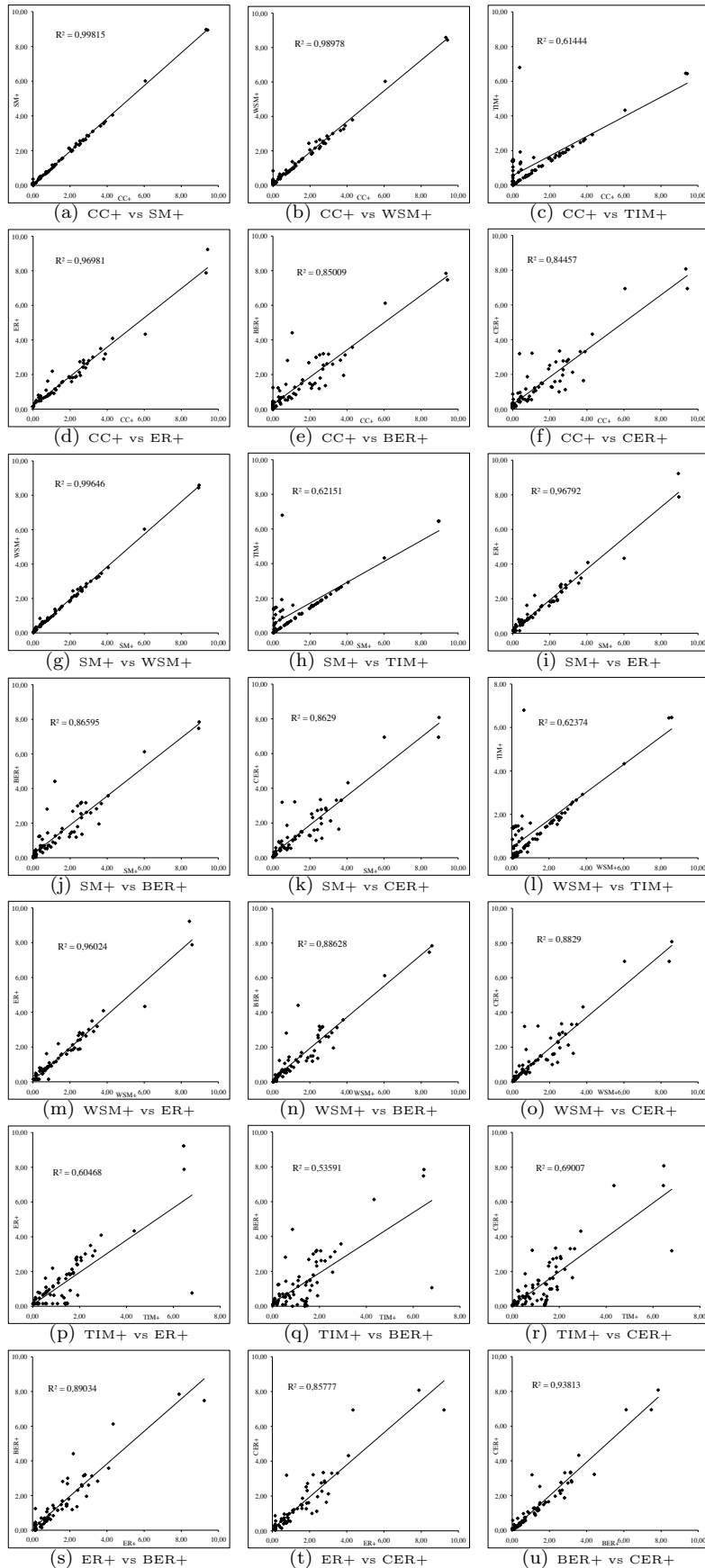


Figure 5.10: Comparison between extended methods applied to the osCommerce.

5.3 Timing Evaluation

In the evaluation of every computational method, an important thing is the execution time. In our case, to study the required time to compute the importance of the entity types of a conceptual schema for the selected methods is a main task because of the execution time can be one of the non-functional requirements to choose between a method or another one.

As in previous sections, to evaluate the execution time of each method we have used the conceptual schema of the osCommerce and the UML metaschema. It is important to note that in the case of the osCommerce we have both the structural and behavioural schemas, therefore we applied the structural one (around 80 entity types) to the base versions of the methods, while the extended ones use the complete conceptual schema (around 350 entity types, including the behavioural part). On the other hand, since the UML metaschema does not have behavioural subschema, we applied to all type of methods the same structural part (around 300 entity types).

To compute the execution time we have placed one timer before the call of each method, and another one after such call. The difference between the measures of these pair of timers results in the execution time. It is worth mentioning that we do not take into account the spended time in the load of each schema, in order to achieve a higher degree of independence of the schema size. This experiment has been done in a Macbook with Mac OS X Tiger v10.4.11, a 2GHz Intel Core 2 Duo processor and 1Gb 667 MHz DDR2 RAM memory. Table 5.4 shows the results obtained by base and extended methods, measured in milliseconds, for both conceptual schemas.

	osCOMMERCE	UML METASHEMA
CC	8	17
SM	20	40
WSM	22	46
TIM	28	98
ER	102	2104
BER	130	1893
CER	150	2886
CC+	38	36
SM+	51	48
WSM+	57	53
TIM+	93	113
ER+	3704	2216
BER+	3711	2093
CER+	4042	2957

Table 5.4: Values in milliseconds (*ms*) of execution time.

Furthermore, Fig. 5.11 shows a plot with the values. It is possible to see that the slower methods are the methods based on link analysis due to compute the importance of entity types they need to solve an equation system by

iterative steps until to reach to the convergence point where the solution is a valid solution.

An interesting behaviour here is the difference between the execution time of the EntityRank (ER) and BEntityRank(BER) methods (in both base and extended cases). We see that although the BER takes into account more information from the schema, it is a little bit faster than the ER. The reason is that BER uses the percentage of attributes for the random jump to reach each entity type (that is the relevance of the first step in the iterative computation required to solve the equation system, as we will explain in the next chapter). Such value is more significant than the used by the ER (the same value for each entity type) and implies a closest way to reach the convergence to a solution for the equation system.

Another important characteristic is that the base versions ER, BER and CER are faster with the osCommerce than with the UML metaschema. As we explained before, it is due to the number of entities of each schema. While to these methods we use the structural part of the osCommerce with over 80 entity types, the number of entity types of the UML metaschema is clearly higher. If we look at the ER+, BER+ and CER+, there the behaviour is different according to we use the complete osCommerce schema with around 350 entity types.

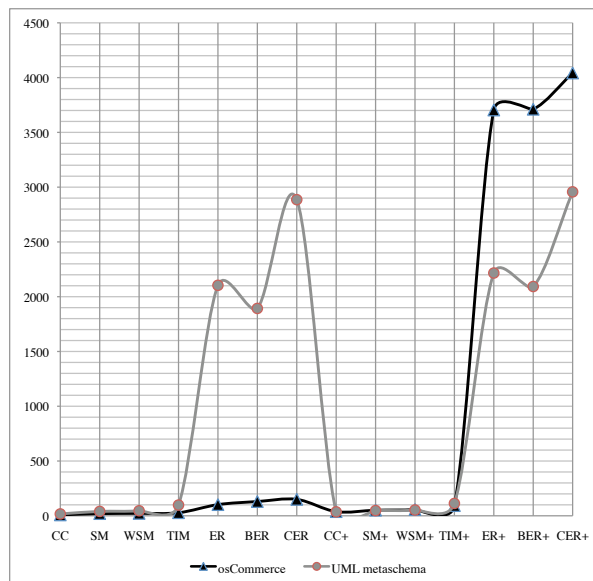


Figure 5.11: Values in milliseconds (*ms*) of execution time.

Another important experiment that is interesting is to compute the average time dedicated for each entity type to compute its relevance value. It is only necessary to divide the previous values by the number of entity types in every case. Table 5.5 shows these values.

As the reader can see in Fig. 5.12, the behaviour is very similar than those

5.3. Timing Evaluation

	OSCOMMERCE	UML METASHEMA
CC	0.09	0.06
SM	0.24	0.14
WSM	0.26	0.16
TIM	0.33	0.35
ER	1.20	7.18
BER	1.53	6.46
CER	1.76	9.85
CC+	0.11	0.12
SM+	0.15	0.16
WSM+	0.17	0.18
TIM+	0.27	0.39
ER+	10.71	7.56
BER+	10.72	7.14
CER+	11.68	10.09

Table 5.5: Values in milliseconds (*ms*) of execution time per entity type.

presented in Fig. 5.11. We can conclude that the methods that are more affected by the size of the conceptual schema are those based in link analysis. It is important to highlight the good results of the methods based on occurrence counting being very similar in both original and extended versions. It implies that although the knowledge used in the extended versions is clearly higher in size, the execution time is not affected. It is different for the link analysis methods, as shown in the case of the osCommerce where the execution time highly increases due to the increment of knowledge taken into account by ER+, BER+ and CER+.

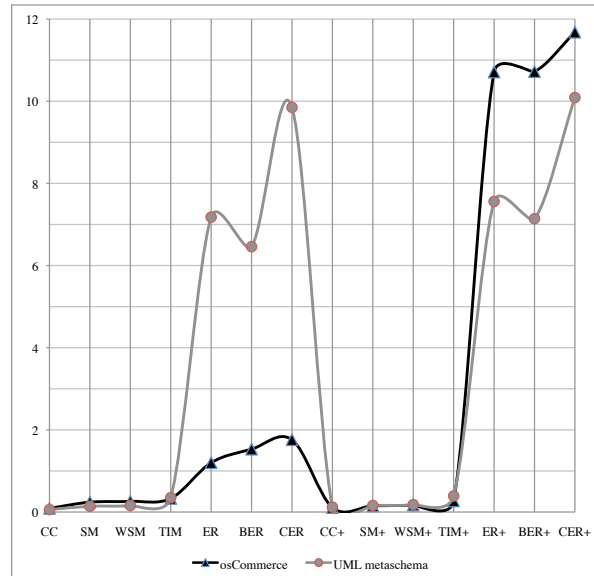


Figure 5.12: Values in milliseconds (*ms*) of execution time per entity type.

5.4 Evaluation of Results

To conclude the chapter, this section presents the results obtained by the original and extended versions of the selected methods once applied to the conceptual schema of the osCommerce and the UML metaschema. Table 5.6 shows the top-10 entity types of the osCommerce, Table 5.7 shows the top-10 entity types of the UML metaschema and, finally, Table 5.8 shows the top-10 event types of the osCommerce obtained by the application of the extended versions of the methods.

Although the results have some differences, it is easy to verify that there exists a set of entity types that appear in almost all the top-10 rankings shown. In the case of the osCommerce, we have Product, Language, Customer, Order, Attribute, TaxZone and OrderStatus as examples of such top relevant entities. On the other hand, the UML metaschema contains a big amount of entity types belonging to the kernel part of such schema (have the prefix Kernel in their name) as for example Classifier, Property, Type, Operation and Element.

Furthermore, as we introduced in earlier chapters, modeling event types of the behavioural schema as entity types implies that the methods for compute the relevance of entity types can be applied to event types. This way we obtain the top-10 rankings shown in Table 5.8. A set of the most relevant event types for the osCommerce are OrderConfirmation, NewProduct, EditProduct and NewCustomerAddress.

The experience obtained from the study done along this chapter allows us to draw some conclusions about which method should be according to our requirements:

- Use a link analysis based method (ER, BER, CER) in its original version in case the behavioural schema were not available to contribute in the importance computing process. Such methods are the ones that better approximate the results obtained using extended versions of the methods.
- Use an occurrence counting based method in its extended version (CC+, SM+, WSM+) in case the behavioural schema were available to contribute in the importance computing process. Such methods give similar results than extended link analysis based ones, with a better execution time.
- Avoid using link analysis based methods if your main requirement is to minimize the execution time and your schemas are big enough.
- Remember that these are only basic orientations. Try to test more than one method with the same schema, if possible, in order to select the one that better adapts to your needs.

5.4. Evaluation of Results

(a) CC vs CC+			(b) SM vs SM+		
	CC	CC+		SM	SM+
1	Product	Language	1	Customer	Product
2	Language	Product	2	Store	Language
3	Store	Customer	3	Product	Customer
4	Customer	TaxZone	4	Language	TaxZone
5	Order	Order	5	MinimumValues	Order
6	Attribute	Attribute	6	Currency	Attribute
7	Currency	OrderStatus	7	Address	OrderStatus
8	OrderStatus	Category	8	PaymentMethod	Category
9	Country	Zone	9	Session	Session
10	TaxZone	Session	10	Banner	Zone

(c) WSM vs WSM+			(d) TIM vs TIM+		
	WSM	WSM+		TIM	TIM+
1	Customer	Product	1	Special	Special
2	Store	Language	2	Customer	Product
3	MinimumValues	Customer	3	Store	Language
4	Product	TaxZone	4	Product	Customer
5	Address	Order	5	PaymentMethod	TaxZone
6	Currency	Attribute	6	Language	Order
7	Language	OrderStatus	7	MinimumValues	Attribute
8	Banner	Category	8	Currency	OrderStatus
9	Session	Session	9	Address	Category
10	PaymentMethod	Zone	10	USPostalService	Session

(e) ER vs ER+			(f) BER vs BER+		
	ER	ER+		BER	BER+
1	Product	Language	1	Product	Product
2	Language	Product	2	Language	Language
3	Store	Customer	3	Customer	Customer
4	Customer	TaxZone	4	Store	Banner
5	Order	OrderStatus	5	Country	TaxZone
6	Country	Order	6	Currency	Country
7	Attribute	Category	7	Address	Zone
8	Currency	Attribute	8	Order	Currency
9	OrderStatus	Country	9	Banner	Order
10	TaxZone	Zone	10	Session	Address

(g) CER vs CER+		
	CER	CER+
1	Product	Product
2	Language	Language
3	Customer	Customer
4	Store	TaxZone
5	Currency	Currency
6	Order	OrderStatus
7	Country	Order
8	Address	Banner
9	OrderStatus	Special
10	Banner	Zone

Table 5.6: Comparison between the top-10 of the methods for the osCommerce.

(a) CC vs CC+			(b) SM vs SM+		
	CC	CC+		SM	SM+
1	BasicActivities_InputPin	BasicActivities_InputPin	1	Kernel_Classifier	BasicActivities_InputPin
2	Kernel_Classifier	Kernel_Property	2	BasicActivities_InputPin	Kernel_Property
3	BasicActivities_OutputPin	Kernel_Classifier	3	BasicActivities_Action	Kernel_Classifier
4	Kernel_ValueSpecification	BasicActivities_OutputPin	4	Kernel_ValueSpecification	BasicActivities_OutputPin
5	BasicBehaviors_Behavior	Kernel_Type	5	Kernel_NamedElement	Kernel_Operation
6	Kernel_Constraint	Kernel_Operation	6	Kernel_Element	Kernel_Type
7	Kernel_Property	BehaviorStateMachines_Transition	7	Kernel_Property	BehaviorStateMachines_Transition
8	Kernel_Operation	Kernel_ValueSpecification	8	BasicActivities_OutputPin	Kernel_ValueSpecification
9	BehaviorStateMachines_State	Kernel_Constraint	9	BasicBehaviors_Behavior	Kernel_NamedElement
10	BasicActivities_ActivityNode	Kernel_Association	10	Kernel_Operation	Kernel_Association

(c) WSM vs WSM+			(d) TIM vs TIM+		
	WSM	WSM+		TIM	TIM+
1	Kernel_Classifier	BasicActivities_InputPin	1	Nodes_Node	Nodes_Node
2	BasicActivities_Action	Kernel_Property	2	Nodes_Device	Collaborations_Collaboration
3	Kernel_NamedElement	Kernel_Classifier	3	Nodes_ExecutionEnvironment	Nodes_Device
4	Kernel_Property	Kernel_Operation	4	Collaborations_Collaboration	Nodes_ExecutionEnvironment
5	Kernel_Element	BasicActivities_OutputPin	5	PackagingComponents_Component	PackagingComponents_Component
6	Kernel_ValueSpecification	Kernel_Type	6	BasicComponents_Component	StructuredClasses_Class
7	BasicActivities_InputPin	Kernel_NamedElement	7	StructuredClasses_Class	BasicComponents_Component
8	Kernel_Operation	BehaviorStateMachines_Transition	8	Kernel_Element	Kernel_Classifier
9	BehaviorStateMachines_State	Kernel_ValueSpecification	9	Kernel_NamedElement	Kernel_Element
10	BasicBehaviors_Behavior	BehaviorStateMachines_State	10	InternalStructures_StructuredClassifier	Kernel_NamedElement

(e) ER vs ER+			(f) BER vs BER+		
	ER	ER+		BER	BER+
1	BasicActivities_InputPin	BasicActivities_InputPin	1	Kernel_Property	Kernel_Property
2	BasicActivities_OutputPin	Kernel_Classifier	2	Kernel_Classifier	Kernel_Operation
3	Kernel_Classifier	BasicActivities_OutputPin	3	BasicBehaviors_Behavior	BasicBehaviors_Behavior
4	Kernel_ValueSpecification	Kernel_ValueSpecification	4	Kernel_ValueSpecification	Kernel_Classifier
5	BasicBehaviors_Behavior	Kernel_Property	5	BasicActivities_InputPin	Kernel_ValueSpecification
6	Kernel_Constraint	Kernel_Operation	6	Kernel_Operation	BasicActivities_InputPin
7	Kernel_Property	BasicBehaviors_Behavior	7	BehaviorStateMachines_State	BehaviorStateMachines_State
8	Kernel_Operation	Kernel_Constraint	8	Kernel_Constraint	Kernel_Parameter
9	Interfaces_Interface	Kernel_Type	9	BasicActivities_OutputPin	Kernel_NamedElement
10	BehaviorStateMachines_State	BehaviorStateMachines_Transition	10	Kernel_Parameter	Kernel_MultiplicityElement

(g) CER vs CER+		
	CER	CER+
1	Kernel_Classifier	Kernel_Classifier
2	Kernel_ValueSpecification	Kernel_Element
3	BasicBehaviors_Behavior	Kernel_NamedElement
4	Kernel_Property	Kernel_ValueSpecification
5	Kernel_Element	Kernel_Property
6	BasicActivities_InputPin	Kernel_Operation
7	Kernel_NamedElement	BasicActivities_InputPin
8	Kernel_Operation	BasicBehaviors_Behavior
9	BehaviorStateMachines_State	Kernel_Parameter
10	Kernel_Constraint	BehaviorStateMachines_State

Table 5.7: Comparison between the top-10 of the methods for the UML metaschema.

5.4. Evaluation of Results

(a) CC+		(b) SM+	
	CC+		SM+
1	OrderConfirmation	1	DomainEvent
2	ExistingCustomerEvent	2	OrderConfirmation
3	ProductDownload	3	ExistingCustomerEvent
4	NewCustomerAddress	4	EditPaymentMethodEvent
5	EditPaymentMethodEvent	5	NewCustomerAddress
6	ExistingProductEvent	6	ProductDownload
7	NewProductAttribute	7	ExistingProductEvent
8	LogIn	8	EditProduct
9	EditProduct	9	NewProductAttribute
10	AddProductToShoppingCart	10	NewProduct

(c) WSM+		(d) TIM+	
	WSM+		TIM+
1	DomainEvent	1	Event
2	OrderConfirmation	2	DomainEvent
3	ExistingCustomerEvent	3	ProductDownload
4	NewCustomerAddress	4	NewCustomerAddress
5	EditPaymentMethodEvent	5	LogIn
6	ExistingProductEvent	6	RestorePreviousShoppingCart
7	ExistingProductEvent	7	EditProduct
8	ProductDownload	8	OrderConfirmation
9	NewProduct	9	EditCustomerAddress
10	NewDownloadableProductAttribute	10	DeleteCustomerAddress

(e) ER+		(f) BER+	
	ER+		BER+
1	OrderConfirmation	1	NewCustomerAddress
2	ExistingCustomerEvent	2	OrderConfirmation
3	EditPaymentMethodEvent	3	EditBanner
4	ProductDownload	4	NewBanner
5	NewCustomerAddress	5	ExistingCustomerEvent
6	ExistingNewsletterEvent	6	EditProduct
7	EditProduct	7	NewProduct
8	ExistingProductEvent	8	ExistingBannerEvent
9	NewProduct	9	ExistingNewsletterEvent
10	NewProductAttribute	10	ProductDownload

(g) CER+	
	CER+
1	DomainEvent
2	ExistingCustomerEvent
3	EditPaymentMethodEvent
4	OrderConfirmation
5	NewCustomerAddress
6	ExistingSpecialEvent
7	ExistingBannerEvent
8	ExistingProductEvent
9	EditBanner
10	NewBanner

Table 5.8: Comparison between the top-10 event types of the methods for the os-Commerce.

Chapter 6

Implementation

One of the main points in scientific research that sometimes is hidden to the audience consists on the implementation of the methods or techniques described in scientific papers, conference proceedings or journals. Obviously, the availability of the implementation in computing is always a good thing because it promotes the research in the area of study and allows a way of check the value of the experiments.

This chapter introduces the implementation we have done of the methods presented in previous chapter in order to apply their functionality to an existing tool. Thus, we provide other researchers the manner to repeat the experiments explained in Chapter 5, and to check the methods with their own schemas without the need of coding them from the scratch. The conceptual schema of the osCommerce and the UML metaschema are contained in our release.

Section 6.1 describes the USE tool [26] as the selected piece of software that allows to work with conceptual schemas. Such tool has been extended with the functionality of create focused views containing the most relevant subset of entity types of a conceptual schema. This behaviour is achieved by the application of the importance computing methods, that are also included in the extended version of the USE tool.

Section 6.2 shows the architecture we have follow in our approach of implementation. The main objectives here are the reusability and independence of the code that implement the explained methods. Section 6.3 explains the process we have follow to compute the link analysis methods and their equation systems. Our approach will be the same as the one by Tzitzikas and Hainaut [54] using linear algebra versions of the methods. Finally, Section 6.4 includes an example of how to use the final version of the extended USE in order to compute the ranking of entity types and the focused views.

6.1 The USE Tool

The USE (UML-based Specification Environment) tool [26] is a system for the specification of information systems implemented in Java. According to its web page¹, it is based on a subset of the Unified Modeling Language (UML)[39]. Roughly, it is an interpreter for a subset of UML and OCL.

To load a conceptual schema inside USE, a specification file has to be written containing a textual description of the schema using features found in UML class diagrams (classes, associations, etc.) and expressions written in the Object Constraint Language (OCL) to specify additional integrity constraints.

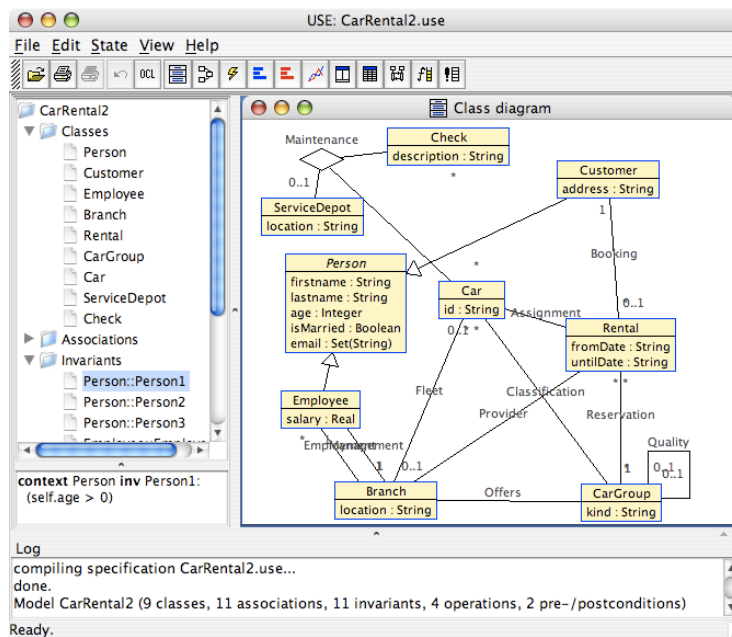


Figure 6.1: Snapshot of the USE tool.

The USE system supports developers in analyzing the model structure (classes, associations, attributes, and invariants) and the model behavior (operations and pre- and postconditions) by generating typical snapshots (system states) and by executing typical operation sequences (scenarios). Developers can formally check constraints (invariants and pre- and postconditions) against their expectations and can, to a certain extent, derive formal model properties. One of the main characteristics of the USE tool is its OCL interpreter. It is possible to specify schema rules in OCL and to validate their syntax and semantics against a schema.

As we will explain in next sections, our main purpose is to include the metrics and methods explained in Chapter 4 into the core of USE.

¹<http://www.db.informatik.uni-bremen.de/projects/USE/>

6.2 The Architecture

One of the main reasons to select USE to implement the methods in is the well-structured source code it has. To understand the code and the different functional areas is an easy task because of the good comments and the division of the components in different packages.

Although we think that USE is a good tool to test the methods, one of the main objectives here is to achieve a sufficient degree of independence between our extension and the USE itself. Thus, we have designed an architecture where the relevance computing methods do not work directly with the USE Java objects but with a middleware piece of code that defines all characteristics required by such methods. Fig. 6.2 shows with a high level of detail the main pieces of our design.

First step consists on to define a set of interfaces to maintain information about the schema elements. Such interfaces must be implemented (including all its operations that are not shown here for the sake of simplicity) for every tool we want to extend. As an example, in the case of Generalization, we have implemented such interface in USE with the Java class `USEGeneralization`. Such class allows us to maintain information about a generalization object of USE but in a way that our implemented methods can understand.

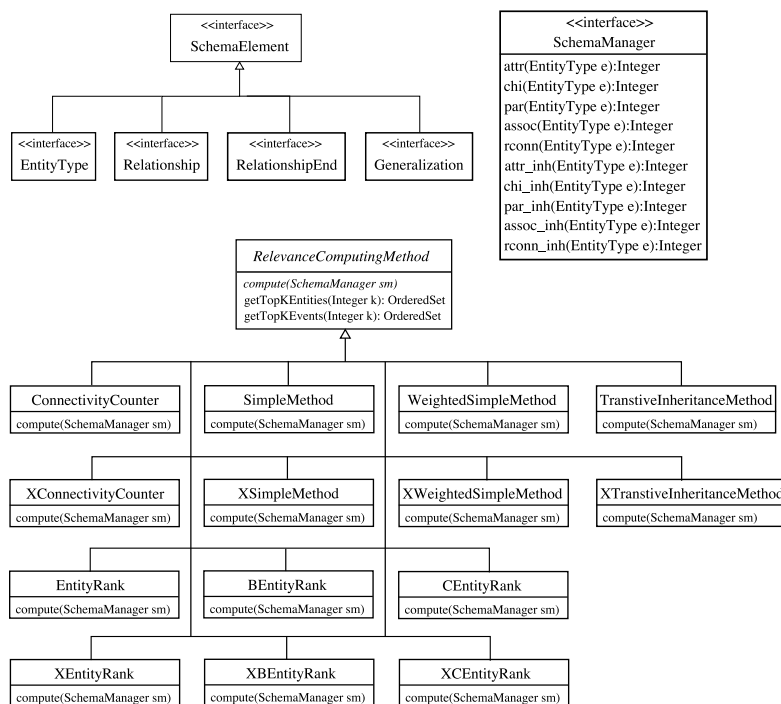


Figure 6.2: Architecture design for the implementation of the relevance computing methods.

The reader can also see that there is another interface to implement in Fig. 6.2. The SchemaManager interface has the task of manage the information of the original conceptual schema (loaded in USE or any similar tool) and to index it in order to obtain the information needed by the metrics used by the methods. It must be implemented according to the tool where the extension is placed. In the case of USE, the implementation of the SchemaManager converts the main elements of the original schema in USE format to instances of the implementation of the schema element interfaces presented before. These objects and the information of the metrics are maintained by the schema manager.

Finally, we observe in Fig. 6.2 the hierarchy of relevance computing methods. These methods are fully implemented and there is no necessary to add more code to them unless new functionalities could be required. The implementation of such methods obtain the information about the conceptual schema to work with through the platform specific implementation of the SchemaManager. Fig. 6.3 shows the steps followed once a request to compute the importance of the entity types has been done.

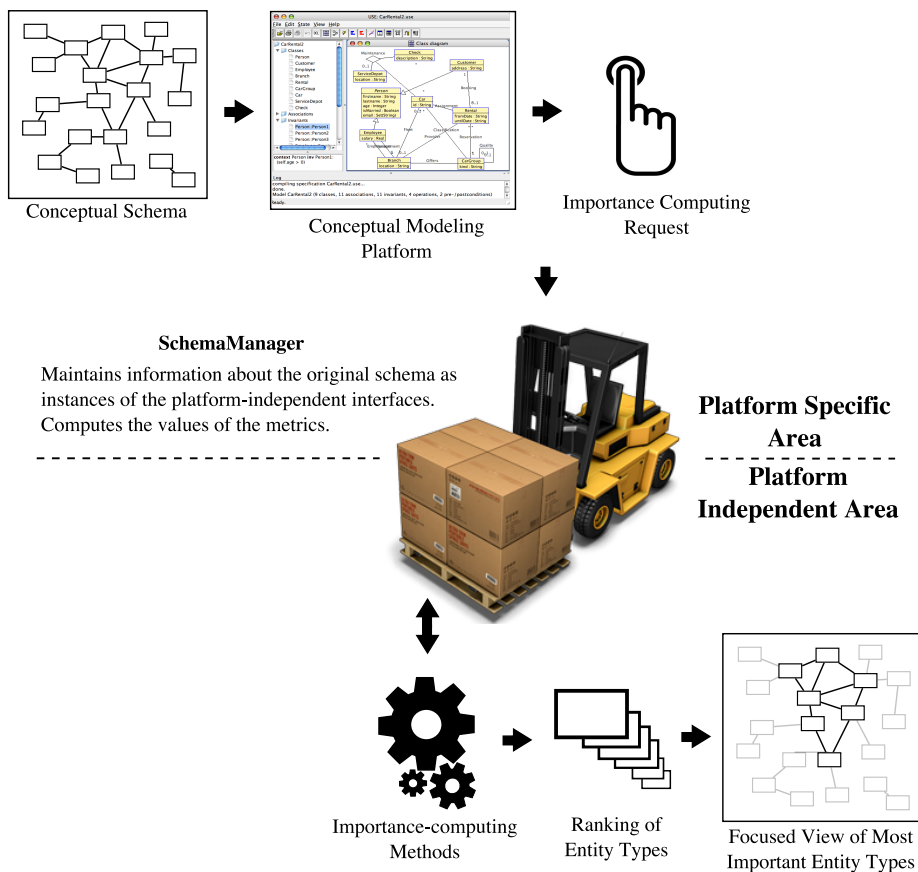


Figure 6.3: Steps to compute the importance of the entity types belonging to a conceptual schema.

The SchemaManager acts as a bridge between the platform specific area (in our case, the USE tool) and the platform independent area (the implementation of the relevance computing methods). This way changes in the way the conceptual modeling platform loads and maintains the information about the conceptual schema implies changes in the SchemaManager, but not in the importance-computing methods.

For the special case of the schema rules, the USE tool creates a tree for each rule according to an ad-hoc structure of expressions similar to the OCL metamodel. To be able to manage and discover the new relationships extracted from the schema rules, the SchemaManager has been extended with a piece of code to visit the trees representing such rules. This mechanism is platform dependent and has to be change for each platform.

6.3 Computing Link Analysis based Methods

The relevance computation of entity types belonging to a conceptual schema can be a very simple process if we choose to apply an occurrence counting based methods. In that case, it consists only in to add the values of each of the metrics taken into account in the method. However, as explained in previous chapters, there exists a set of methods based on link analysis that require a more complex computation process to be applied.

In this section we introduce new versions of the ER, BER, CER and their extensions in order to be computed using a mix of linear algebra concepts and Markov chain theory.

6.3.1 The Power of Linear Algebra

Chapter 4 introduced the formal definition for the link analysis based methods. As explained there, such methods produce an equation system that must be solved in order to obtain the relevance value for each entity type. In a sense, we affirm that the relevance of an entity type $e \in \mathcal{E}$ has a proportional relation with the addition of the relevance values of those entity types that are linked with e . Look at Fig. 6.4:

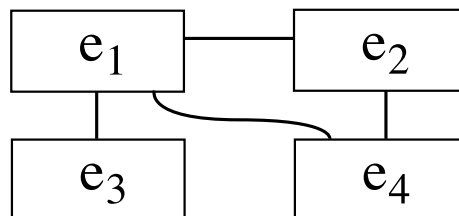


Figure 6.4: Example of simple conceptual schema.

Therefore we have:

$$\begin{aligned} I(e_1) &= K(I(e_2) + I(e_3) + I(e_4)) \\ I(e_2) &= K(I(e_1) + I(e_4)) \\ I(e_3) &= K(I(e_1)) \\ I(e_4) &= K(I(e_1) + I(e_2)), \end{aligned}$$

where K is a proportionality constant and each $I(e)$ is the importance or relevance of the entity type e .

Such equation system can be transformed into a matrix equation:

$$\begin{pmatrix} I(e_1) \\ I(e_2) \\ I(e_3) \\ I(e_4) \end{pmatrix} = K \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} I(e_1) \\ I(e_2) \\ I(e_3) \\ I(e_4) \end{pmatrix}$$

Now we can rewrite the previous matrix equation: we call i to the importance vector. The $n \times n$ matrix (in this example $n = 3$) of the system is the M associated to the schema graph. So that we can write the importance assignment that we want as a solution of

$$M i = \lambda i.$$

Now we take λ as a proportionality constant ($\lambda = 1/K$). Therefore the problem has become into an eigenvalues and eigenvectors problem. Our importance vector i is an eigenvector of the M matrix, which contained the structure of the schema graph.

To obtain such eigenvector, the *Perron-Frobenius* theorem asserts that a real square ($n \times n$) matrix with positive entries has a unique largest real eigenvalue $u > 0$ and that the corresponding eigenvector v has strictly positive components [10]. Therefore, our work consists on decompose the matrix M to obtain its eigenvalues and eigenvectors. Then, according to the theorem, the eigenvector v associated to the largest eigenvalue u will be the desired importance vector i .

Finally, to compute the importance of entity types, we redefine the link analysis based methods formalization into the formalization of a Markov chain. We define for each method a transition matrix T . According to the Markov chain theory, the stationary distribution of importance is the principal eigenvector of the transpose of the matrix T .

To decompose such matrix we use JAMA², which is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. It is meant to provide sufficient functionality for routine problems, packaged in a way that is natural and understandable to non-experts. Concretely, it provides the eigenvalue decomposition of both symmetric and non-symmetric square matrices.

²<http://math.nist.gov/javanumerics/jama/>

Our implementation obtains the vector of eigenvalues and selects the one with a greater positive value. Then, we obtain the matrix of eigenvectors and select the one associated to the previous selected eigenvalue. Such eigenvector contains for each component the importance of one entity type.

$$i = \begin{pmatrix} I(e_1) \\ I(e_2) \\ \vdots \\ I(e_n) \end{pmatrix}$$

6.3.2 EntityRank (ER)

EntityRank is based on a Markov chain on the entity types with a transition matrix

$$T_{ER} = q \cdot U + (1 - q) \cdot M,$$

where $A[e_i, e_j] = |\{e_j \mid e_j \in \text{conn}(e_i)\}|$ and M is the stochastic matrix from A , that is, the normalization of A with each row summing to 1. It is possible to say that A is the adjacency matrix of the entities and relationships of the conceptual schema.

Furthermore, U is the transition matrix of uniform transition probabilities where $U[e_i, e_j] = 1/|\mathcal{E}|$. And q is the same random jump factor as explained in the original definition of the EntityRank method. In this case, all entity types have the same initial probability due to we use the matrix U . As we explained in Chapter 5, it implies a little bit more execution time to reach the convergence of the method.

The resulting matrix for the example of Fig. 6.4 would be

$$T_{ER} = q \cdot \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix} + (1 - q) \cdot \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix},$$

and the decomposition of eigenvectors and eigenvalues must be done with the transpose of T_{ER} .

If we choose $q = 0.15$, we have the following matrix to decompose:

$$\begin{pmatrix} 0,0375 & 0,4625 & 0,8875 & 0,4625 \\ 0,32083333 & 0,0375 & 0,0375 & 0,4625 \\ 0,32083333 & 0,0375 & 0,0375 & 0,0375 \\ 0,32083333 & 0,4625 & 0,0375 & 0,0375 \end{pmatrix}$$

Then, we obtain the eigenvectors and eigenvalues shown in the Table 6.1:

		EIGENVALUES			
		1	-0,62	0,19	-0,42
ENTITY TYPES	ASSOCIATED EIGENVECTORS				
E1	0.6988	0.8933	0.4339	0.0000	
E2	0.4686	-0.2423	-0.5332	-0.7071	
E3	0.2694	-0.4086	0.6324	-0.0000	
E4	0.4686	-0.2423	-0.5332	0.7071	

Table 6.1: Eigenvalues and associated eigenvectors

The greater positive eigenvalue is 1, then we select its associated eigenvector ($I_{ER}(e_1) = 0.6988$, $I_{ER}(e_2) = 0.4686$, $I_{ER}(e_3) = 0.2694$, $I_{ER}(e_4) = 0.4686$) as the resulting importance vector. If we normalize the components to sum 1, we have the next ranking:

ENTITY TYPE	RELEVANCE
E1	0,36
E2	0,25
E4	0,25
E3	0,14

Table 6.2: Ranking of entity types for the example of Fig. 6.4 using the linear algebra version of the ER.

6.3.3 BEntityRank (BER)

BEntityRank is based on a Markov chain on the entity types with a transition matrix

$$T_{BER} = q \cdot B + (1 - q) \cdot M,$$

where $A[e_i, e_j] = |\{e_j \mid e_j \in \text{conn}(e_i)\}|$ and M is the stochastic matrix from A , that is, the normalization of A with each row summing to 1. It is possible to say that A is the adjacency matrix of the entities and relationships of the conceptual schema.

Furthermore, B is the transition matrix of of transition probabilities where $B[e_i, e_j] = \frac{|\text{attr}(e_j)|}{|A|}$. B is stochastic, meaning that should be normalized in order to each row sums 1. And q is the same random jump factor as explained in the original definition of the EntityRank method. In this case, each entity type has a different initial probability due to we use the matrix B taking into account the attributes. Thus, the more attributes an entity type has, the more relevant is.

The resulting matrix for the example of Fig. 6.4, if we imagine that $attr(e_1) = 1$, $attr(e_2) = 0$, $attr(e_3) = 4$ and $attr(e_4) = 2$, would be

$$T_{BER} = q \cdot \begin{pmatrix} 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \end{pmatrix} + (1 - q) \cdot \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix},$$

and the decomposition of eigenvectors and eigenvalues must be done with the transpose of T_{BER} .

6.3.4 CEntityRank (CER)

CEntityRank is based on a Markov chain on the entity types with a transition matrix

$$T_{CER} = q_1 \cdot B + q_2 \cdot H + (1 - q_1 - q_2) \cdot M,$$

where $A[e_i, e_j] = |\{e_j \mid e_j \in conn(e_i)\}|$ and M is the stochastic matrix from A , that is, the normalization of A with each row summing to 1. It is possible to say that A is the adjacency matrix of the entities and relationships of the conceptual schema.

Furthermore, B is the transition matrix of transition probabilities where $B[e_i, e_j] = \frac{|attr(e_j)|}{|A|}$. And q is the same random jump factor as explained in the original definition of the EntityRank method. In this case, each entity type has a different initial probability due to we use the matrix B taking into account the attributes. Thus, the more attributes an entity type has, the more relevant is.

And H is the transition matrix taking into account generalizations where

$$H[e_i, e_j] = \begin{cases} 1, & \text{if } e_j \in gen(e_i) \\ 0, & \text{otherwise} \end{cases},$$

and H is stochastic, meaning that should be normalized in order to each row sums 1.

The resulting matrix for the example of Fig. 6.4, if we imagine that $attr(e_1) = 1$, $attr(e_2) = 0$, $attr(e_3) = 4$ and $attr(e_4) = 2$, and that $gen(e_1) = e_2$ would be

$$T_{CER} = q_1 \cdot \begin{pmatrix} 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \\ 1/7 & 0 & 4/7 & 2/7 \end{pmatrix} + q_2 \cdot \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\ + (1 - q_1 - q_2) \cdot \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 0 & 1/2 \\ 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{pmatrix},$$

and the decomposition of eigenvectors and eigenvalues must be done with the transpose of T_{CER} .

6.3.5 EntityRank Extended (ER+)

The EntityRank extended method is based on a Markov chain on the entity types with a transition matrix

$$T_{ER+} = q \cdot U + (1 - q) \cdot (M + R),$$

where U and M are the same than in case of the T_{ER} and the transition matrix R takes into account the relationships extracted from the schema rules of the conceptual schema. Formally, $R[e_i, e_j] = |\{e_j \mid e_j \in rconn(e_i)\}|$. Roughly, $R[e_i, e_j]$ contains the number of participations that e_i has with e_j , in the extracted relationships from the schema rules. It is important to note that R is stochastic, meaning that should be normalized in order to each row sums 1.

6.3.6 BEntityRank Extended (BER+)

The BEntityRank extended method is based on a Markov chain on the entity types with a transition matrix

$$T_{BER+} = q \cdot B + (1 - q) \cdot (M + R),$$

where B and M are the same than in case of the T_{BER} and the transition matrix R takes into account the relationships extracted from the schema rules of the conceptual schema. Formally, $R[e_i, e_j] = |\{e_j \mid e_j \in rconn(e_i)\}|$. It is important to note that R is stochastic, meaning that should be normalized in order to each row sums 1.

6.3.7 CEntityRank Extended (CER+)

The CEntityRank extended method is based on a Markov chain on the entity types with a transition matrix

$$T_{CER+} = q_1 \cdot B + q_2 \cdot H + (1 - q_1 - q_2) \cdot (M + R),$$

where B , H and M are the same than in case of the T_{CER} and the transition matrix R takes into account the relationships extracted from the schema rules of the conceptual schema. Formally, $R[e_i, e_j] = |\{e_j \mid e_j \in rconn(e_i)\}|$. It is important to note that R is stochastic, meaning that should be normalized in order to each row sums 1.

6.4 User Manual

This section presents how work with the modified version of the the USE tool that includes the implemented extension to compute the relevance of entity types. First we generally show the way to execute the USE environment and to load conceptual schemas. Then, a description of the three main functionalities of our extension is presented to conclude the section and the chapter.

6.4.1 Working with USE

The extended version of the USE tool can be downloaded from the web site of the author of this thesis³. We call this version as USE 2.4.0 WIC to indicate that is the same distribution than official USE 2.4.0⁴ but including the importance computing methods (WIC – With Importance Computing).

First step consists on to open the USE environment. To do this task, the user must have installed a Java Runtime Environment 1.5 or greater⁵. Once it is accomplished, to start USE a windows user must open the base directory of the distribution (downloaded from the author’s web) and access to the *bin* directory. There, the user must double click on the *use.bat* file and a new USE window should appear as follows:

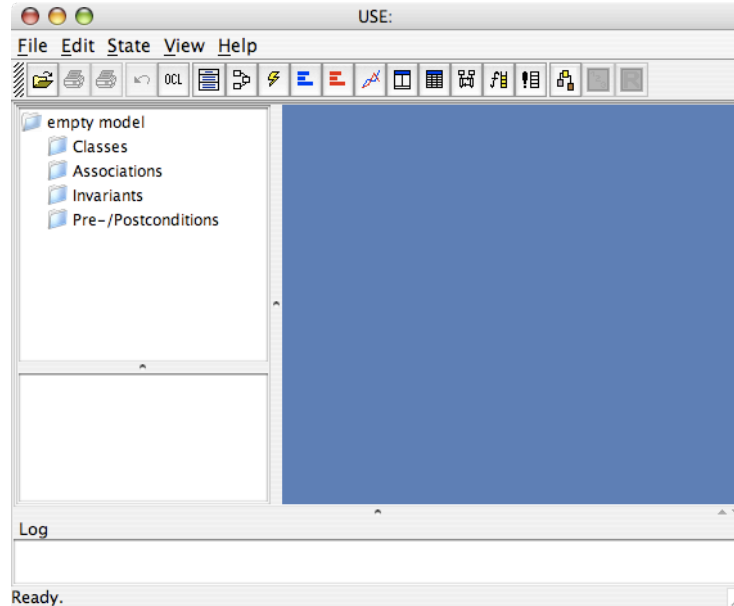


Figure 6.5: USE main window.

³<http://www.lsi.upc.edu/~avillegas/Resources/use-2.4.0-wic.zip>

⁴<http://www.db.informatik.uni-bremen.de/projects/USE/>

⁵<http://www.java.com/en/download/manual.jsp>

The Unix-based user (Linux, MacOS X, ...) must open a terminal and execute the next command within the same *bin* directory:

```
$use-2.4.0-wic\bin\>sh use  
or also  
$use-2.4.0-wic\bin\>./use
```

The next step is to load a specification of a conceptual schema in USE format. To do that, the user must click on the first icon of the toolbar (a typical open folder image) or also go to the **File->Open specification...** command of the Menu bar.

Then, after selecting the desired *.use* file, the log message area shows the characteristics of the loaded schema. To access to the conceptual schemas of this thesis (the *osCommerce* and the UML metaschema) the user must access the *examples* directory of the base distribution, and then go to the *AVillegas-Thesis* directory. Look at Fig. 6.6.

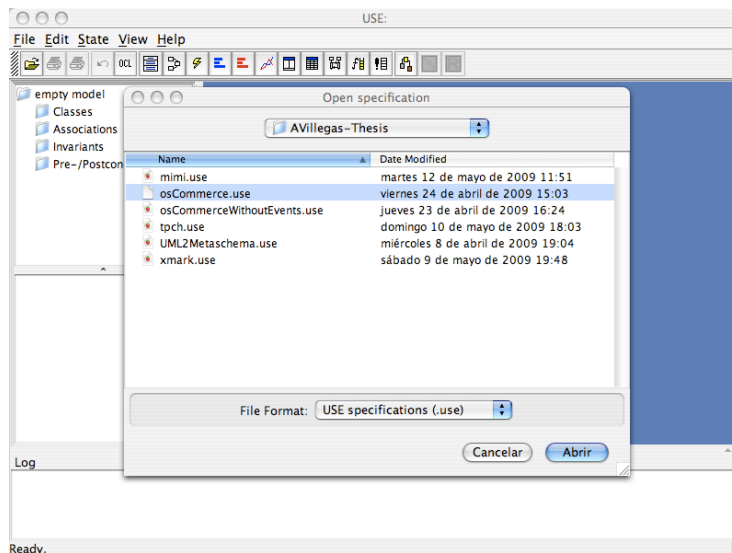


Figure 6.6: Open a conceptual schema in USE.

The current specification format for to describe conceptual schemas for the USE environment is described in the *use-documentation.pdf* document that is included in the base directory of the USE 2.4.0 WIC distribution. Such document also explains in more detail the main functionalities of the USE tool. As an example, Fig. 6.7 shows the class diagram view of the *osCommerce* schema loaded in USE. To open such view, the user must click on the sixth icon of the menu bar (the one with an UML class picture).

As the user can see, the class diagram shown in Fig. 6.7 is not useful to comprehend the main aspects of the *osCommerce* schema. By the way, we

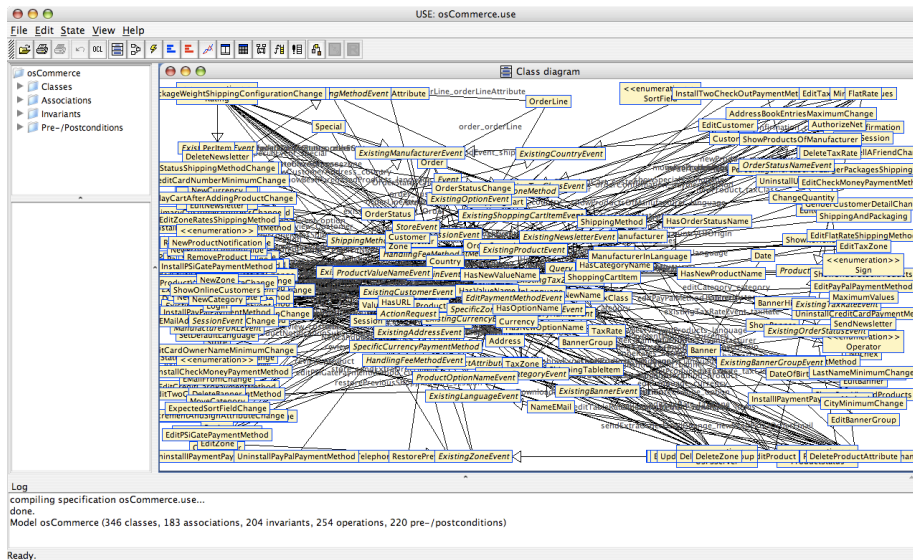






Figure 6.7: Class diagram view of the osCommerce schema in USE.

only have shown the entities and relationships, hiding another elements like multiplicities of associations, attributes or role names.

6.4.2 Compute the Relevance of Entity Types

Now, we present how to compute the relevance of the entity types of a conceptual schema loaded into the USE environment. The most important buttons of the toolbar here are   . Such icons are new from the original version of USE and have been included in our extension.

First button in the toolbar of USE () allows users to select one of the 14 methods to compute the importance of entity types and to apply it to the current loaded schema. Fig. 6.8 shows a screenshot of that selection.

It is important to note that USE does not identify event types as a special kind of entity types. Therefore, to separate the events from the entities is not provided here. Of course, a possible solution could be to mark as event each entity type that directly or indirectly is a descendant of another entity type named Event. However, we do not implement it due to a conceptual schema could have an entity type with such name but not being an event type.

Once the method is selected, the user must choose a text file where the complete results of the method are stored in order to be consulted in more detail. For the case of occurrence counting methods, this output file contains the values of each entity type for all the metrics taken into account in the method. On the other hand, link analysis methods produce only the final relevance value into

the output file.

Furthermore, the result output file is written as a table with each column separated with tabs in order to simplify the import process into other tools like Excel, R or any other data mining environment.

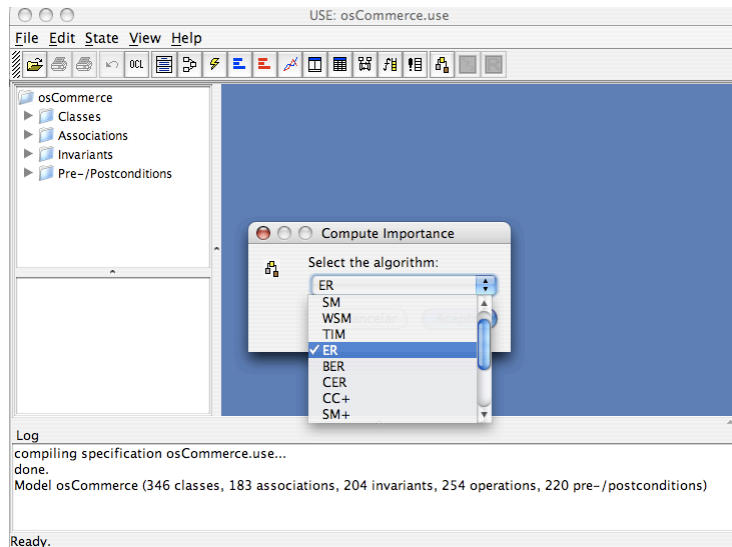


Figure 6.8: Selection of the importance computing method in USE.

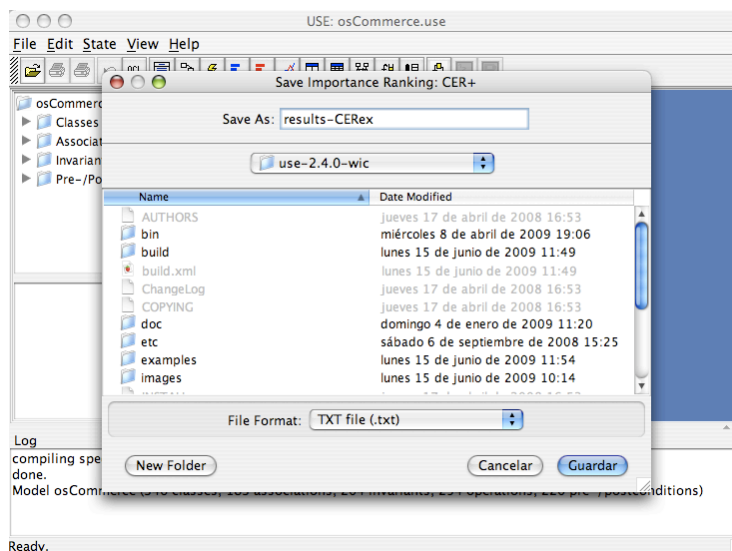



Figure 6.9: Saving the results of an importance computing method in USE.

6.4.3 Visualization of Results

Once we have one of the importance computing methods applied to the schema in USE, next step consists on the visualization of the results. Our extension to the original USE environment allows to both view a ranking list of the entity types and a reduced view of the whole class diagram of the schema.

The second button we have included in the original tool bar () is disabled until a method is applied to the schema. Afterwards, the button becomes active and therefore a ranking list view as the one shown in Fig. 6.10 appears.

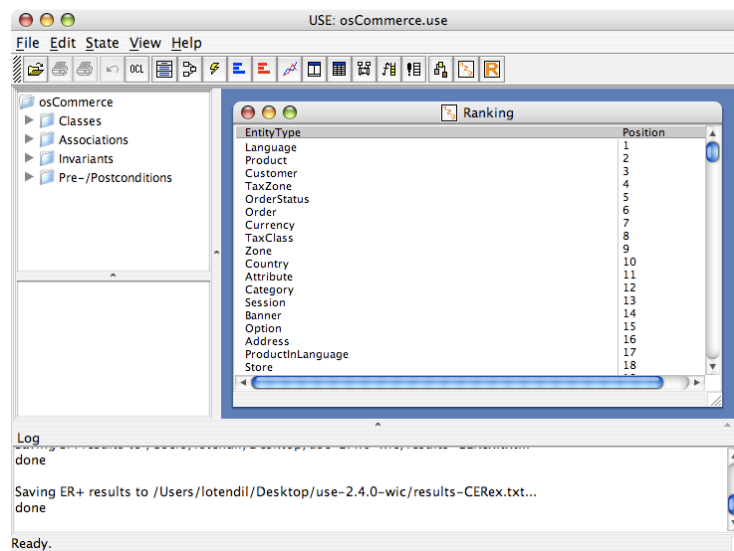



Figure 6.10: Ranking list of the osCommerce in USE.

Finally, the third button included in the extended USE () is probably the most useful one. It is also disabled until a method is applied to the schema. Once enabled, it provides a reduced top-20 view of entity types. We have chosen 20 entity types because is a number of entities that in major cases can be placed into a single page. Look at the example in Fig. 6.11.

It is important to note here that once we have the top-20 entity types selected, all other types are filtered. We only show those relationship types whose participants are all included in the top-20. Furthermore, the initial view only shows the entities and relationships.

The user can reduce the number of entity types shown by selecting those entity types to hide and using the contextual menu (right click on the class diagram view with some entities selected) that provides a set of options to modify the information shown in the diagram.

To conclude the section, we show in Fig. 6.12 the top-10 entity types of the

6.4. User Manual

osCommerce once the extended version of the EntityRank (ER+) method is applied. We have successfully hide the entities in the position 11 to 20 and also we have chosen to see the information about attributes, multiplicities, and role names in associations. The user can compare this view with the one in Fig. 6.7 to be aware of the benefits of filtering.

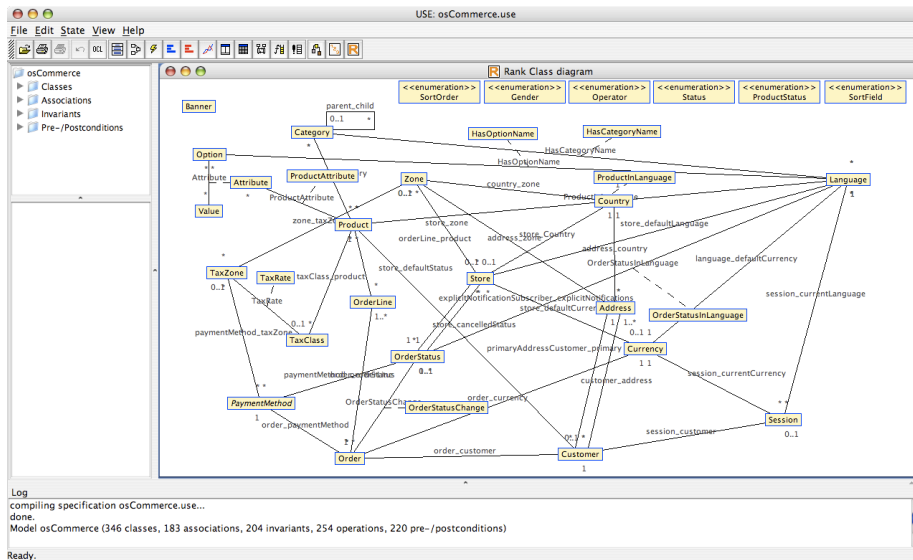


Figure 6.11: Top-20 entity types of the osCommerce in USE.

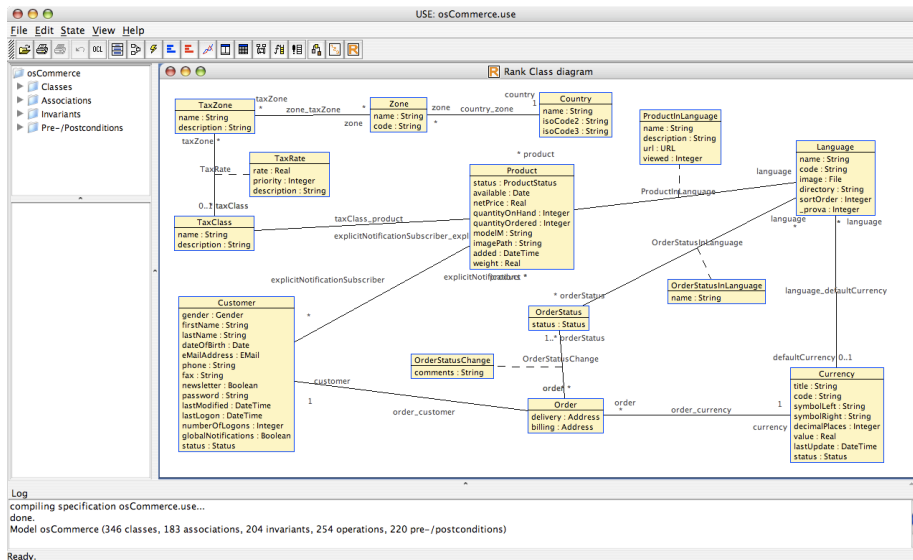


Figure 6.12: Top-10 entity types of the osCommerce in USE.

Chapter 7

Conclusions

This chapter states some conclusions about the research topic of the relevance computing on conceptual modeling. The aim is to summarize the main contributions of this document and the state of the art in this research field.

In the same way we did with conceptual schemas, this chapter gives a focused view of the main contents of the thesis, in order to check the fulfillment of the main objectives presented in the first chapter. Afterwards, we also show some possible extensions to the work presented along the document that could be completed in future research.

7.1 Conceptual Modeling

We have explained the importance of conceptual modeling in the development process of information systems. This subject has increased its relevance from consisting into merely documentation activities to be a key set of tasks in the analysis and design phases of the development cycle.

Furthermore, one of the main reasons for this important growth in conceptual modeling interest is the increasing need for information management. Since conceptual schemas, as the key artifacts in conceptual modeling, have to maintain big amounts of concepts, relationships and so many other characteristics of an specific information domain, end-users are getting more uncomfortable with such schemas.

As we have seen in some chapters, conceptual modeling needs the aid of specific techniques to reduce the amount of information that stakeholders of the information system have to manage. Humans are not able to process big amounts of information because their perception and cognitive systems are not prepared to do that. We presented some references that confirm humans cannot deal with such quantities of data.

So there exists a problem in conceptual modeling when the conceptual schemas get big enough to overtake the limit of human perception. However, there also exists a set of methods to reduce the information to deal with. The main objective of the thesis was to study such methods and to deliver whether they are useful or not. In addition, another purpose was to establish a set of extended methods to take into account the whole information included in conceptual schemas.

7.2 State of the Art

The research field this thesis is included in covers a huge amount of subjects. Some of them are visualization of information, graph theory, or data mining. As the reader can observe, such areas are so many general and our intention was to study a more specific problem. Concretely, the thesis was centered in computing the importance of entity types because entities are the main element in conceptual schemas. Nevertheless, we presented a complete state of the art in the whole research field in order to get a correct overview of the main contributions other researchers have submitted until now.

One of the main methods to reduce the amount of information were the clustering methods. As early explained, these kind of methods group together similar elements into sets of similar information, also known as clusters. The research contributions in this area was big, although their application in real cases was not so large.

On the other hand, there were another methods following a different approach. It consists on hide the irrelevant information putting the focus in the most important concepts. To know which are the most relevant concepts, this techniques make ranking lists. This way, end users are able to get an overview of the whole with a reduced effort. This filtering methods were less studied than clustering, so they have become into the main research point of this document.

It is also important to note that existing contributions had some deficiencies. Major research articles in this field do not include cognitive justification, having based their approaches or solutions in principles different than of human information processing. Furthermore, a big limitation in most of the contributions was the lack of automation. Only a little part of the whole proposals contain notes about the implementation the authors have done, and only an even more reduced part includes the algorithms, or at least a pseudo-code version of them.

But the most important problem about the methods that try to reduce the information overload in conceptual schemas is their lack of empirical testing. Most of the proposals are only limited to argue or affirm that the solution they provide has a better result than others. The cause of this deficiency is the subjectivity of the different ways of evaluate the benefits of the methods. Major evaluations are based on the advises of experts or *oracles* in the field of the conceptual schema. Thus, the most similar solution to the solution given by an *oracle* is chosen as the better one. However, to share the same *oracle* in two or

more different articles is not an option and, therefore, the comparisons are not easily done.

Another thing that increases the limitations of the proposals is the use of only a little subset of the whole conceptual schema to discover the most important entity types. Since a conceptual schema contains not only a structural schema, but also a behavioural schema full of event types and new relationships that conform a valuable amount of information, to be limited in the use of only a subset of the structural schema (entities, relationships and, sometimes, attributes) is not a good thing.

The power that the OCL language provides must be taken into account and the central part of this thesis was dedicated to discover new ways of uncover information of the conceptual schema traditionally forgotten from the point of view of the importance computation process.

7.3 Thesis Contribution

The main contribution this thesis provides is the extension of a set of representing methods in the literature with new knowledge traditionally forgotten. Our approach takes into account not only a subset of the structural schema but also the behavioural schema and all the schema rules. Furthermore, we convert graphical constraints like the multiplicities of associations into new schema rules thanks to the OCL language. We use that extra knowledge because of our main idea is to avoid the need of having the collaboration of experts in the field of the conceptual schema. Concretely, we propose a more objective way to compute the importance of schema elements, and particularly entity types, based on the amount of information dedicated to each entity type. Therefore, the more knowledge an entity type has in the conceptual schema, the higher degree of relevance it owns.

As we explained in previous chapters, the usage of the information provided by the schema rules and the event types and pre- and postconditions of the behavioural schema gives a component of subjectivity to the final relevance value of the entity types. This is due to the requirements engineers that adapt the requirements of an information system not only into the structural schema defining concepts and the relationships between them, but also including behaviour information about the actions and events the information system must and must not perform. Thus, from the point of view of the users, their requirements are included in the conceptual schema and, afterwards, in the importance computing process.

Our contribution also includes the implementation of the methods explained in the document in order to be tested and evaluated with our conceptual schemas or others. This way, the researchers with interest in this are can repeat our experiments to check the correctness of them or also imagine new situations that worth study. We do not close the code we have used, but we offer it as a free open source contribution that can be extended to include new methods or

to be adapted to other modeling environments. Furthermore, we chose USE as the main environment of our extension due to its simplicity and because to add new functionality is an easy job.

Another important aspect is the brief explanation we did about the limits on human perception. This way, we decided to reduce the whole conceptual schemas to a focused view which can fulfill in a single page. Although there were some indications in the literature that state a number in the range of 5 to 10 concepts as the limit of human information processing capacity, we implemented the application of the methods giving a top-20 diagram, which is reasonable because there exists the possibility of hiding elements. Thus, the user has the last word to decide the size he/she wants to work with.

Some other lack in the literature is the use of little schemas to check the proposals in the field of conceptual modeling in the large. Our contribution here is the proof of our methods with two large conceptual schemas, including a real business case (the osCommerce) and a more theoretical case (the UML metaschema). Both schemas contain around 300 entity types, so we consider them as clear cases of large schemas that are correct to be used in our experiments.

Finally, we have shown an experimental evaluation with the original methods and the proposed extensions. Concretely we have concluded that link-analysis methods are the best methods when the behavioural schema or the schema rules written in OCL are not available, but such methods are slower than occurrence counting ones due to its recursive definition. In addition to it, we showed some advice to choose the best method in each situation that state occurrence counting methods are good when the knowledge we dispose is bigger. Thus, having big amounts of knowledge implies use occurrence counting methods while having less knowledge is a situation where the link-analysis methods obtain better results.

After the study done along this research work we conclude that the more amount of information we have available in the conceptual schema, the better are the results of the importance computing process. So, this is a reason to spend some time describing not only the structure but also the behaviour in conceptual modeling.

7.4 Future Work

The realization of this research work has produce some new ideas that were not included in this thesis. Anyway, it is worth mentioning some of them as future work to complete the research in this field. Probably the most relevant task we have to improve is the evaluation of the methods with more conceptual schemas. As explained, to have the opportunity of work with real conceptual schemas at our disposal is not a real situation. Organizations take a special care of their schemas as a main artifact in the business strategy, therefore to convince such organizations to let us to look at their schemas is not an easy task.

However, we have tested our methods with two large schemas which is not a common situation in most of the existing contributions. Furthermore, a good future research work could be to create a repository of free large conceptual schemas in the same format to be accessed by everyone interested in this topic. Also, a benchmark should be described in order to have a place where to compare the results obtained by the methods and, therefore, select those that are the best ones.

Another future task could be the relevance computing of the elements that were not valued in the scope of this document. We mean attributes, relationships and also schema rules. This way the reduced overview of the conceptual schema could contain not all the attributes, but only the most relevant ones, and in the same way select the most important relationships and hide the irrelevant. We know that although we compute the importance of entity types and (with a minor degree) event types, other elements must be taken into account to be evaluated.

In addition to the previous future tasks, an interesting future activity should be the study and improvement of the visualization techniques used to increase the usability of conceptual schemas. Carrying out the state of the art in this topic, we looked at some references but we realize that there exists a need of more in-deep work in order to profit the benefits that good and usable interfaces could provide to the work with conceptual schemas. As we explained, there is a big amount of visualization techniques that are not really taken into account in real cases that require to be studied and implemented.

Finally, we have discovered new methods in the field of community structure detection that could be added to our selected methods. Such approaches compute the centrality of the edges in a graph and then pass their importance to the nodes that are connected to them. This idea could be adapted to the scope of conceptual schemas processing the centrality of relationship types and then converting such values into importance for the entity types.

Bibliography

- [1] ABREU, F. Metrics for Object Oriented Software Development. In *3rd International Conference on Software Quality, Lake Tahoe, Nevada, EUA* (1993).
- [2] AKOKA, J., AND COMYN-WATTIAU, I. Entity-relationship and object-oriented model automatic clustering. *Data & Knowledge Engineering* 20, 2 (1996), 87–117.
- [3] BAEZA-YATES, R., RIBEIRO-NETO, B., ET AL. *Modern information retrieval*. Addison-Wesley Harlow, England, 1999.
- [4] BARONI, A., AND ABREU, F. Formalizing object-oriented design metrics upon the UML meta-model. In *Brazilian Symposium on Software Engineering, Gramado-RS, Brazil* (2002).
- [5] BARONI, A., AND E ABREU, F. A formal library for aiding metrics extraction. In *International Workshop on Object-Oriented Re-Engineering at ECOOP* (2003).
- [6] BARONI, A., AND E ABREU, F. An OCL-based formalization of the MOOSE metric suite. *Proc. of QUAOOSE* (2003).
- [7] BARONI, A. L. Formal definition of object-oriented design metrics. Master's thesis, Vrije Universiteit Brussel, 2002.
- [8] BAUERDICK, H., GOGOLLA, M., AND GUTSCHE, F. UML 2.0 Validation Results in Form of an EXCEL, PDF, and USE File. University of Bremen. 2004. [ftp://ftp.informatik.uni-bremen.de/local/db/papers/uml2004/ocl_uml2.\[xls|pdf|use\]](ftp://ftp.informatik.uni-bremen.de/local/db/papers/uml2004/ocl_uml2.[xls|pdf|use]).
- [9] BAUERDICK, H., GOGOLLA, M., AND GUTSCHE, F. Detecting OCL Traps in the UML 2.0 Superstructure: An Experience Report. In *UML* (2004), T. Baar, A. Strohmeier, A. M. D. Moreira, and S. J. Mellor, Eds., vol. 3273 of *LNCS*, Springer, pp. 188–196.
- [10] BERMAN, A., AND PLEMMONS, R. *Nonnegative matrices in the mathematical sciences*. Society for Industrial Mathematics, 1994.
- [11] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems* (1998), Elsevier Science Publishers B. V., pp. 107–117.

- [12] CAMPBELL, L., HALPIN, T., AND PROPER, H. Conceptual schemas with abstractions making flat conceptual schemas more comprehensible. *Data & Knowledge Engineering* 20, 1 (1996), 39–85.
- [13] CASTANO, S., ANTONELLIS, V. D., FUGINI, M. G., AND PERNICI, B. Conceptual schema analysis: Techniques and applications. *ACM Trans. Database Syst.* 23, 3 (1998), 286–332.
- [14] CHEN, P. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)* 1, 1 (1976), 9–36.
- [15] CHIDAMBER, S., AND KEMERER, C. MOOSE: Metrics for object oriented software engineering. In *Workshop on Processes and Metrics for Object-Oriented Software Development (OOPSLA'93), Washington DC, EUA, September (1993)*.
- [16] COCKBURN, A., KARLSON, A., AND BEDERSON, B. A review of overview+detail, zooming, and focus+ context interfaces. *ACM Computing Surveys* 41, 1 (2008).
- [17] COWAN, N. Evolving conceptions of memory storage, selective attention, and their mutual constraints within the human information processing system. *Psychological Bulletin* 104, 2 (1988), 163–191.
- [18] DEREMER, F., AND KRON, H. Programming-in-the large versus programming-in-the-small. In *Proceedings of the international conference on Reliable software table of contents (1975)*, ACM New York, NY, USA, pp. 114–121.
- [19] ESTIVILL-CASTRO, V. Why so many clustering algorithms: a position paper. *ACM SIGKDD Explorations Newsletter* 4, 1 (2002), 65–75.
- [20] FELDMAN, P., AND MILLER, D. Entity model clustering: Structuring a data model by abstraction. *The Computer Journal* 29, 4 (1986), 348–360.
- [21] FRANCALANCI, C., AND PERNICI, B. Abstraction levels for entity-relationship schemas. *Lecture Notes in Computer Science (1994)*, 456–456.
- [22] GANDHI, M., ROBERTSON, E., AND GUCHT, D. Leveled entity relationship model. *Lecture Notes in Computer Science (1994)*, 420–420.
- [23] GEERTS, F., MANNILA, H., AND TERZI, E. Relational link-based ranking. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30 (2004)*, VLDB Endowment, pp. 552–563.
- [24] GENERO, M., POELS, G., AND PIATTINI, M. Defining and validating metrics for assessing the maintainability of entity-relationship diagrams. *Faculteit Economie en Bedrijfsjunde Hoveniersberg – Working Paper Series 11 03/199 (2003)*.
- [25] GENERO, M., POELS, G., AND PIATTINI, M. Defining and validating metrics for assessing the understandability of entity-relationship diagrams. *Data & Knowledge Engineering (2007)*.

- [26] GOGOLLA, M., BÜTTNER, F., AND RICHTERS, M. Use: A uml-based specification environment for validating uml and ocl. *Sci. Comput. Program.* 69, 1-3 (2007), 27–34.
- [27] HAN, J., AND KAMBER, M. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
- [28] JAESCHKE, P., OBERWEIS, A., AND STUCKY, W. Extending ER model clustering by relationship clustering. *Lecture Notes in Computer Science* (1994), 451–451.
- [29] KATIFORI, A., HALATSIS, C., LEPOURAS, G., VASSILAKIS, C., AND GI-ANNOPOULOU, E. Ontology visualization methods—a survey. *ACM Computing Surveys* 39, 4 (2007).
- [30] KLEINBERG, J. Authoritative sources in a hyperlinked environment. *Journal of the ACM* 46, 5 (1999), 604–632.
- [31] KLEPPE, A., WARMER, J., AND BAST, W. *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2003.
- [32] KOSARA, R., MIKSCH, S., AND HAUSER, H. Focus+ context taken literally. *IEEE Computer Graphics and Applications* 22, 1 (2002), 22–29.
- [33] LINDLAND, O. I., SINDRE, G., AND SØLVBERG, A. Understanding quality in conceptual modeling. *IEEE Software* 11, 2 (1994), 42–49.
- [34] MAROIS, R., AND IVANOFF, J. Capacity limits of information processing in the brain. *Trends in Cognitive Sciences* 9, 6 (2005), 296–305.
- [35] MILLER, G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychol. Rev* 63 (1956), 81–97.
- [36] MOODY, D., AND FLITMAN, A. A decomposition method for entity relationship models: a systems theoretic approach. In *International Conference on Systems Thinking in Management* (2000), pp. 462–469.
- [37] MOODY, D. L., AND FLITMAN, A. A Methodology for Clustering Entity Relationship Models – A Human Information Processing Approach. In *ER 1999* (1999), J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, and E. Métails, Eds., vol. 1728 of *LNCS*, Springer, pp. 114–130.
- [38] OBJECT MANAGEMENT GROUP (OMG). *Object Constraint Language Specification (OCL), version 2.0*, May 2006.
- [39] OBJECT MANAGEMENT GROUP (OMG). *Unified Modeling Language (UML) Superstructure Specification, version 2.2*, February 2009.
- [40] OLIVÉ, A. *Conceptual Modeling of Information Systems*. Springer-Verlag, 2007.

- [41] OLIVÉ, A., AND CABOT, J. A research agenda for conceptual schema-centric development. In *Conceptual Modelling in Information Systems Engineering*, S. B. John Krogstie, Andreas Lothe Opdahl, Ed. Springer Verlag, 2007, pp. 319–334.
- [42] OLIVÉ, A., AND RAVENTÓS, R. Modeling events as entities in object-oriented conceptual modeling languages. *Data & Knowledge Engineering* 58, 3 (2006), 243–262.
- [43] PAPAZOGLOU, M. P. Unraveling the semantics of conceptual schemas. *Commun. ACM* 38, 9 (1995), 80–94.
- [44] RICHTERS, M., AND GOGOLLA, M. OCL: Syntax, semantics, and tools. *Lecture Notes in Computer Science 2263* (2002), 42–68.
- [45] SALTON, G. Automatic text processing. *Addison-Wesley Series In Computer Science* (1988), 450.
- [46] SALTON, G., AND MCGILL, M. *Introduction to modern information retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1986.
- [47] SHOVAL, P., DANOCH, R., AND BALABAM, M. Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. *Requirements Engineering* 9, 4 (2004), 217–228.
- [48] SHOVAL, P., DANOCH, R., AND BALABAN, M. Hierarchical ER Diagrams (HERD)-The Method and Experimental Evaluation. *Lecture Notes in Computer Science* (2003), 264–274.
- [49] STREIT, A., PHAM, B., AND BROWN, R. Visualization support for managing large business process specifications. *Lecture Notes in Computer Science 3649* (2005), 205.
- [50] TAVANA, M., JOGLEKAR, P., AND REDMOND, M. An automated entity-relationship clustering algorithm for conceptual database design. *Information Systems* 32, 5 (2007), 773–792.
- [51] TEOREY, T. J., WEI, G., BOLTON, D. L., AND KOENIG, J. A. Er model clustering as an aid for user communication and documentation in database design. *Commun. ACM* 32, 8 (1989), 975–987.
- [52] TORT, A., AND OLIVÉ, A. The osCommerce Conceptual Schema. Universitat Politècnica de Catalunya. <http://guifre.lsi.upc.edu/OSCommerce.pdf>, 2007.
- [53] TZITZIKAS, Y., AND HAINAUT, J. On the visualization of large-sized ontologies. In *Proceedings of the working conference on Advanced visual interfaces* (2006), ACM New York, NY, USA, pp. 99–102.
- [54] TZITZIKAS, Y., AND HAINAUT, J.-L. How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms). In *ER 2005* (2005), L. M. L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, and O. Pastor, Eds., vol. 3716 of *LNCS*, Springer, pp. 144–159.

- [55] TZITZIKAS, Y., KOTZINOS, D., AND THEOHARIS, Y. On ranking rdf schema elements (and its application in visualization). *J. UCS* 13, 12 (2007), 1854–1880.
- [56] VARGA, R. *Matrix iterative analysis*. Springer, 2000.
- [57] WARMER, J., AND KLEPPE, A. *The object constraint language: precise modeling with UML*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.
- [58] YU, C., AND JAGADISH, H. V. Schema summarization. In *VLDB (2006)*, U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, Eds., ACM, pp. 319–330.

