UDC 004.652

Sabine Müllenbach[1], Ph.D., Professor of Computer Sciences Faculty,
E-mail: sabine.muellenbach@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-0392-0334
Lore Kern-Bausch[1], Ph.D., Professor of Computer Sciences Faculty,
E-mail: lore.kern-bausch@hs-augsburg.de, ORCID https://orcid.org/0000-0003-3401-1333
Matthias Kolonko[1], Diplom-Wirtschaftsinformatiker (FH), assistant of Faculty of Computer Science,
E-mail: matthias.kolonko@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-8296-1758
[1]Augsburg University of Applied Sciences (AUAS), An der Hochschule 1, Augsburg, Germany, 86161
Tel. +49 (821) 5586-0

CONCEPTUAL MODELING LANGUAGE AGILA MOD

Abstract. *Modeling of data structures has always been an important topic in the discussions of software engineering practice. Recently, the idea of conceptual modeling has lost importance in these discussions. The fact that research in this area has not been pushed further a lot for the last decade can be considered as an evidence. However, this concept has great potential. Especially the idea of creating a paradigm agnostic model depicting facts of the real world – the so called "Universe of Discourse" – instead of concrete data structures following a certain logical data model makes it so powerful and valuable. Hence, it deserves further research to find best practices to utilize conceptual modeling effectively. The problems that discouraged software engineers from making use of conceptual modeling is that the models are hard to understand. Creating them is time-consuming, other stakeholders do not know what to do with them and creating the final data structures requires an additional process step. After all, it is mostly perceived as too expensive in time and money without creating an appropriate value. In this article, the existing approaches are examined to find out their weaknesses and the reasons why they did not gain a broader acceptance. Therefore, the important requirements that a conceptual modeling language has to meet for practical fielding are determined. Furthermore, the concepts of semantic modeling languages are examined. Using semantics instead of mere structural discussions simplifies access and understanding for non-IT stakeholders. It helps to check the validity of the created data structures against the demands of the real business. In the further course, the concept of semantically irreducible sentence modeling will be discussed which can act as a bridge between semantic and conceptual modeling. With the results of these discussions, the conceptual modeling language AGILA MOD is presented. This modeling language bases on the idea of depicting semantically irreducible sentences as graphical model. By this, it can act as a common platform all project participants can agree upon building the bridge between IT implementation and business requirements. The models can be created from semantically irreducible sentences and they can be read backwards into semantically irreducible sentences making this language easy to understand for all project participants. AGILA MOD is therefore intended to be as easy as possible to get started without a lot of learning efforts. Hence, it bases on the well-known Entity-Relationship language in a simplified variant. A few additional constructs are added that also refer to well-known modeling techniques reducing the efforts of learning new elements nearly to zero. The derivation of AGILA MOD models to a logical model is done by following simple derivation rules making it less time consuming and hence less cost-intensive. This language shall act as a basis for further research targeting towards the new logical models of NoSQL as well as creating a comprehensive framework automating the derivation as much as possible. Additionally, the possibility of making use of polyglot persistence with this approach and the creation of a convenient API shall be considered in future research.*

Keywords: *Database; Conceptual Modeling; Domain mode; Semantic data modeling; Entity-Relationship; Paradigm Agnosticism*

Introduction

In the past years, conceptual data modeling as a general approach has not been evolving a lot further any more. Nowadays, modeling basically has settled on the techniques of Entity-Relationship approaches or UML. Current research basically focuses on specialized scenarios and environments where a general approach may be inappropriate or they already concern the logical level. Practically, it can be experienced that creating a comprehensive data model in a project has become rather unpopular and seldom as it is perceived as an additional task with no real benefit or even as double work and hence too expensive and not worth the effort. However, it is often the complexity of the data that lets projects fail or at least suffer from heavy delays due the created data structure being highly imperformant and computing-intensive.

The idea of this article is not to invent another wheel so users have to learn yet another modeling language, but to find a way to apply the idea of a general conceptual data model as easy as possible and by this make it easily understandable and applicable. Taking into account the concepts of semantic data modeling as well, this approach shall be eligible to act as an interface between business experts and IT experts.

The result should be a model both sides can understand and agree upon on the one hand. On the other hand, this model should still be formalized in a way that it can be easily used as basis to derive its contents into the correct information structures for the IT.

Formulation of the problem

The general approach of data modeling like it is presented in [1] comprises three steps: a conceptual model which is independent of the technical data structure, a logical model which depicts the technical structures due to a selected structural paradigm [1; e.g. relational or hierarchical structures] and a physical model that applies the logical structures to a certain vendor specific instance including all physical attributes to run the database efficiently. Logical and physical model shall be derived directly from the predecessor by a defined ruleset. The real brainwork remains with the conceptual model.

This general approach is widely accepted and can be found until today in relevant literature [1; 2; 3]. In the wider context of project execution, a conceptual model is suitable to be used in the development process as a common platform to acquire a common sense on the development topic between developers and business experts. Such a model could e.g. then be part of the functional specification document customer and contractor agree upon. In addition to data modeling, it is widely used to create a single information space for effective enterprise management [24].

However, to convince the audience to make use of this approach, the entry should be as simple as possible and it should be universally applicable to minimize resistance or even rejection.

Therefore, it is necessary to define a modeling syntax that meets three goals:

1) Understandable by both, IT and business experts, so the model can act as a common platform.

2) Easy syntax to avoid learning efforts.

3) Paradigm agnostic to not anticipate the decision of the target environment.

Analysis of existing scientific achievements and publications

The desired solution needs to be accepted by IT experts, i.e. according to the second goal, they need to be able to master the modeling language quickly or they are even already familiar with it. We can find basically two modeling languages that have settled nowadays: UML and Entity-Relationship.

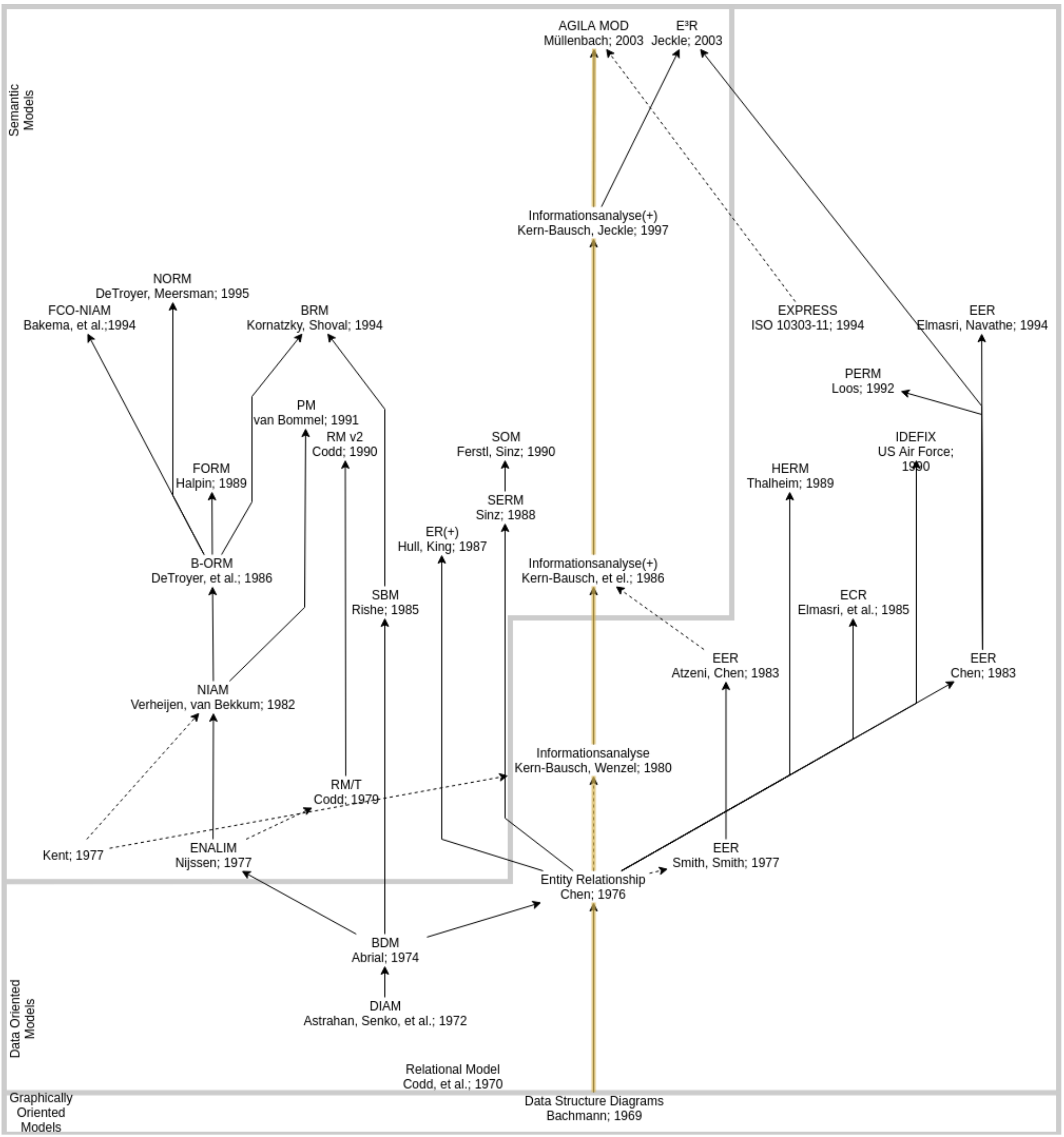
The development of the Unified Modeling Language (UML) started in the 1990s and has developed until today to the version 2.5 which was published in 2013[4; ch.1.4]. This language is a comprehensive framework for software modeling containing structural modeling elements that are of course capable of solving the task to model data structures. It can also be considered that UML is widely accepted among IT experts and hence can be assumed familiar for them. However, UML runs contrary to the first goal of being understandable by non IT experts. The variety of syntax primitives can be confusing at times and especially the structural elements are inspired by technical terms of IT[1, e.g. classes, components or packages] as its main goal is to model “systems” [4; p.21] and not the real world. Hence, UML is hard to understand for non IT experts even though one of its premises is “understandability” [4; p.21]. Furthermore, UML has a strong notion of the object-oriented paradigm and by this runs contrary to the third goal as well. In the end, UML seems to be rather inadequate as a basis for creating the desired approach.

Entity-Relationship (ER) on the other hand came up already in 1976 by Peter Chen [5] and was developed further during the following years by several researchers (Fig. 1). Various constructs have been added to the syntax calling the results “EER”, [1; ch.4] or E^{3}R [6]. Like UML, ER is widely accepted today for modeling data structures in the IT business. It can be found in various literature that concentrate on modeling of data structures. [1; 7; 8]. Hence, ER can be supposed to be familiar among the IT business. Still, ER and most of its derivate target on just representing data structures in a more or less abstract fashion. The notion of a semantic model was only brought up by Hull and King in 1987 with “ER (+)” [17]. Furthermore, the syntax primitives of ER as we know it today are indeed not so numerous and complex like in UML. Still, it can take non-IT experts some time to understand an ER model when they are confronted with one. Though ER has been brought up in combination with relational structures and still is considered in conjunction with it by most people nowadays, ER has become paradigm agnostic as we can see in [1] where it is used as conceptual language and several logical models can be derived from it. Hence, ER in its current forms meets at least goal two and three.

Of course other approaches have also been developed, but never gained as much acceptance as UML or ER. EXPRESS [9] is one example. In the first place, it was developed as a pure lexical representation with a fixed format. It is constructed as a full description framework including data types and structural properties capable to describe highly complex structures and dependencies between them. To ease readability, a graphical representation, “EXPRESS-G”, was also created which contains only a subset of the original possibilities to describe the structures. But with respect to its powerful abilities it is again very complex and hard to understand. Hence, it does not support neither of the formulated goals except being paradigm agnostic. Another example is IDEF1X [10] which suffer from the same peculiarities as EXPRESS and hence, are also not applicable to meet the formulated goals.

Concerning the first goal, the desired solution needs to have the ability to be understandable by all participants, i.e. people who are not necessarily familiar with the technical terms of IT. The most common comprehension among humans is of course natural language. Hence, it seems rather logical to seek for the solution in the area of semantic models. The philosophy behind these models goes beyond the mere representation of structures in a technical sense. They follow the idea of representing the reality as it is utilizing natural language in one way or another [11].

Currently, the Resource Description Framework (RDF; [12]) and the Web Ontology Language (OWL; [13]), both developed under the W3C organization, are the most known representatives concerning semantic description languages and have received already a very high recognition in the IT world. However, these languages have a very technical background and are not meant to be read by some non-IT audience. They are rather an approach to encode semantic meaning from natural language using facilities like XML and URIs so that machines can utilize it, e.g. for artificial intelligence. It is not their intention to act as a link to normal humans. Hence, these technologies are way too technical for meeting the first goal.


Semantic Models
Data Oriented Models
Graphically Oriented Models
AGILA MOD Müllenbach; 2003
E³R Jeckle; 2003
Informationsanalyse(+) Kern-Bausch, Jeckle; 1997
NORM DeTroyer, Meersman; 1995
FCO-NIAM Bakema, et al.;1994
BRM Kornatzky, Shoval; 1994
EXPRESS ISO 10303-11; 1994
EER Elmasri, Navathe; 1994
PERM Loos; 1992
PM van Bommel; 1991
RM v2 Codd; 1990
SOM Ferstl, Sinz; 1990
IDEFIX US Air Force; 1990
FORM Halpin; 1989
HERM Thalheim; 1989
SERM Sinz; 1988
ER(+) Hull, King; 1987
B-ORM DeTroyer, et al.; 1986
Informationsanalyse(+) Kern-Bausch, et al.; 1986
SBM Rishe; 1985
ECR Elmasri, et al.; 1985
EER Atzeni, Chen; 1983
EER Chen; 1983
NIAM Verheijen, van Bekkum; 1982
Informationsanalyse Kern-Bausch, Wenzel; 1980
RM/T Codd; 1979
Kent; 1977
ENALIM Nijssen; 1977
EER Smith, Smith; 1977
Entity Relationship Chen; 1976
BDM Abrial; 1974
DIAM Astrahan, Senko, et al.; 1972
Relational Model Codd, et al.; 1970
Data Structure Diagrams Bachmann; 1969
Fig. 1. Development lines of data modeling techniques Solid edges represent direct derivations, dashed edges represent influences on the development (adapted from [6]; ch.3.1.3; p.74)

A similar example for semantic languages is Gellish [14]. From its basic idea, it is very similar to OWL, but focuses much more on the idea of being a usable not only by IT experts. The basic idea is to describe everything in most simple sentences of the form “subject predicate object” (e.g. “Augsburg lies in Bavaria”). In Gellish it is defined as “left-hand term”, “relation type” and “right-hand term”. Gellish comes with a predefined and comprehensive set of elements containing natural language terms understandable by every human. Still, it allows adding additional elements as necessary at any time. The built sentences are saved as so called “facts”.

All elements are represented in natural language, but get assigned a unique ID simultaneously. By this, Gellish allows to create dictionaries in every language to translate these facts without effort. At the same time facts get assigned a unique ID themselves enabling them to become part of other facts as well. This makes Gellish extremely powerful and still comprehensible by non-IT experts.

The idea of building simple sentences that are comprehensible by everyone is charming and can be a step towards the desired approach. Still, Gellish does not meet the needs completely as the vast set of elements makes it again complicated which contradicts the second goal.The former professor for databases and data modeling of Augsburg University of Applied Sciences (AUAS), Lore Kern-Bausch, developed with Bernd Wenzel in 1980 her own approach simply called “Informationsanalyse” (i.e. “information analysis”) [18]. It was inspired by the original Entity Relationship approach [5] and got further developed in 1986 and 1997 [19; 20]. In fact, this approach introduces another graphical modeling syntax aside from the already well-known ER notation. This is definitely objectionable and contradicts the first and the second goal of finding an appropriate approach. But during its development, this approach received a very fascinating notion of semantic modeling. The authors introduced the method of semantically irreducible sentence building to their syntax. The idea behind this is – similar to Gellish – to build the simple most sentences in natural language that can't be divided into several single sentences without losing parts or all of the semantical meaning behind the original sentence. Every of these semantically irreducible sentences shall then be expressed graphically in the conceptual model with the syntax primitives of the approach of “Informationsanalyse”. By doing so, the graphical model can be created from natural language and natural language can be read from this model reversely. It enables IT experts to just talk to the business experts and make them “read” the model to double check if the model is correct by simply checking the proposition of every sentence. This concept is eligible to meet the first and the second goal as it becomes possible to communicate between IT and business experts and it is paradigm agnostic as the sentences that are translated to graphical notation simply represent facts of the real world independently of any target system environment.

The survey shows that many approaches already exist dealing with the problem of conceptual modeling more or less. All of them have benefits to some of the defined goals for the desired approach. But still, all of them run short in at least one of these goals.

Research methods

AGILA MOD as it is taught today at Augsburg University of Applied Sciences is a resumption of the work of Bernd G. Wenzel and Lore Kern-Bausch titled “Informationsanalyse” [18; 19]. The development has been continued during the teaching period of Lore Kern-Bausch and subsequently Sabine Müllenbach incorporating the experience of working with student teachings and bachelor, master and PhD thesis (Fig. 1).

Presentation of the main research material

The presented approach is called AGILA MOD. Though the name resembles the concepts of “agile development” (cf. [15]), it is more a coincidence in the first place. AGILA stands for “Automatic Generators for Information representation, Logical DB structures and Applications” MOD simply stands for “modeling language”. AGILA is supposed to be a whole framework enabling to create data structures in a generic fashion based on a conceptual model. In this article, the developed modeling language as the absolute basis shall be presented.

The idea of AGILA MOD is to recombine certain elements from the before mentioned approaches to meet the stipulated goals. The starting point builds the concept of the “Informationsanalyse” approach creating a conceptual model by translating natural language as semantically irreducible sentences into graphical notation. This concept aligns perfectly with the first goal of understandability and also with the third goal of paradigm agnostic. But instead of creating a new syntax, the basis of AGILA MOD syntax is ER which aligns to some extent the second goal as the learning efforts are quite low due to the fact that ER is widely known by IT experts. To fully support this goal, the decision was made to reduce the syntax primitives as much as possible. Hence, AGILA MOD uses a subset of the syntax primitives of ER. To support the third goal even more, the wording is slightly changed to set the focus more on the idea of modeling the real world or at least a part of it.(the so called “Universe of Discourse” (UoD)) This philosophy shall be kept in mind by all participants of the modeling process when creating a data model.

Starting from this minimal subset, a few additional elements have been added or adapted from the before mentioned approaches, basically to illustrate certain aspects more in detail than it could be with the base syntax.

Basic syntax elements

The first syntax primitive of AGILA MOD is the "Object Type". It represents a group of real world objects of the same type and is represented by a rectangle with a solid border and the name of the object type inside. In ER it is called an Entity. It was renamed as the term "Object Type" is easier to understand for non-IT experts. There is a distinction between lexical and non-lexical object types. Lexical object types can be represented directly as data in an IT system while non-lexical object types cannot. This distinction depends on semantics in the first place and does not reflect in the model or notation immediately. Examples for non-lexical object types are "Person", "Car" or "Room" whereas lexical object types might by "Name", "Salary" or "Birthday". The concept of lexical and non-lexical object types is adapted from the NIAM modeling language where "LOT" and "NOLOT" are explicitly described [23; ch.4.5].

The second syntax primitive is the "Association Type". It comprises all associations between two or more object types under a certain premise. It is represented by a rhombus with a solid border and the name of the association type inside denoting the premise of the association. An association type must not stand alone it needs at least two participating object types which are connected to the association type via a straight solid edge. Each of these edges needs to be annotated with the role that the instances of the connected object type play in every association. The roles used must be unique for that association type (i.e. there must not be two participants for an association type using the same role name). Furthermore, the edge must be annotated with a cardinality stating how often every instance of the connected object type has to be part of an association at the minimum and maximum. For the boundaries applies: $\{min,max \mid min \in \mathbb{N}_0 \wedge max \in \mathbb{N} \wedge 0 \leq min < +\infty \wedge 0 < max \leq +\infty \wedge min \leq max\}$. The boundaries of the cardinality are notated "<min>:<max>". A maximum boundary of $+\infty$ will be notated as an asterisk ("*"). This differs slightly from Elmasri/Navathe (cf. [1, p.113 bottom]) where the notation is "(<min>:<max>)" and $+\infty$ is represented by the letter "N". The brackets have been omitted for simplicity reasons during modeling while using an asterisk instead of "N" prevents confusions. In the past, students experienced that when using paper and pencil for their models, a badly written 'n' could be misinterpreted as "1" which leads to completely wrong assertions in the model (especially during examinations). An asterisk does not cause such effects. Mentioning a role may be omitted if the role is identical to the name of the object type. It is recommended however to always mention a role for clear expression and the deliberate decision for using the object type in this role.

With these two syntax primitives sentences can already be graphically represented like in

Fig. *2*.

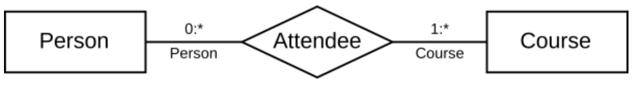


Fig. 2: Association type

This model can be read backwards: "A person can participate in arbitrary courses (but does not have to participate in any)". It also states: "A course needs to be attended by at least one person". Hence, it becomes clear that every association type contains as many facts as participants because it needs to be read from the perspective of every participant.

For association types, there exists only one restriction: If there is a participant with a maximum boundary of "1", the association type must only have exactly two participants. If there were more than two participants, the model would not be semantically irreducible anymore.

Fig. *3* shows such a problematic association type. Reading this backwards results in: "A person has a mobile phone and an office phone". But this sentence can be split into two sentences without losing any information in it: "A person has a mobile phone. A person has an office phone". Hence, the original sentence is semantically reducible and can be equivalently modeled as in [14]

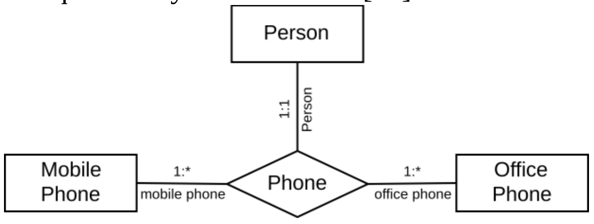


Fig. 3. Erroneous association type. Resulting sentences are not semantically irreducible

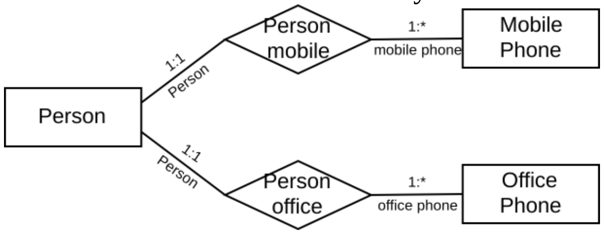


Fig. 4. Correct model

The association type does not impose any further restrictions. It is especially not of relevance what participants are used. For example, it is allowed to let the same object type participate more than once in an association type.

Fig. 5 shows a simple example illustrating that this is not a special case and can occur quite often: The boss of an employee is an employee himself. An employee can have a boss ("can" because the CEO on top won't have one.). And an employee can be boss of arbitrary employees. This example also clarifies why the roles of the participants are vital. An employee must not participate more than once with the same role in this association type.

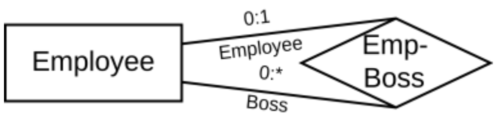


Fig. 5. An object type participating twice in an association type

Naming conventions

Moreover, a special kind of association type was introduced. It has only two participants, both having an upper boundary of "1" (Fig. 6). It implies that for every instance of the first object type there is at most one instance of the second object type assigned and vice versa. This describes the state of a mutual identification between these object types. For this, a simplified association type element was added called a "naming convention". It simply has no label inside, roles of the participants are omitted and the cardinality is only written when it should be "0:1" (otherwise it is "1:1" by default). This abbreviated form of an association type eases the recognition of such identifying connections on first sight by any reader of an AGILA MOD model. The experience with students over the last two decades also showed that it enables modelers to realize an identification more quickly, because an association type having "1:1" on both sides is salient in the model making the modeler reconsider if this identification is really on purpose. Hence, using this abbreviation consequently improves the quality of the model.

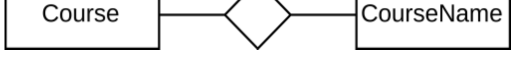


Fig. 6: Naming convention

Objectification

One of the most powerful, still simple constructs of AGILA MOD is the concept of objectification. It enables the modeler to elevate any association type to an object type. By this, a so called "structured object type" is created that can be used like any other object type to describe the universe of discourse.

Fig. 7 shows the example of

Fig. *2* enhanced by an application date using objectification. As in the original example already suggested, the combination of a person and a course can be considered an attendee. From a semantic point of view, it is pretty obvious that an attendee can be perceived as an object type as well. Hence, objectifying this association type enables the modeler to reuse that association type as an object type to model further semantically irreducible sentences where the attendee acts as an object type. In this example the new sentence is: "An attendee has an application date". An association type should only be objectified if needed as an object type. Otherwise, the objectification is redundant.

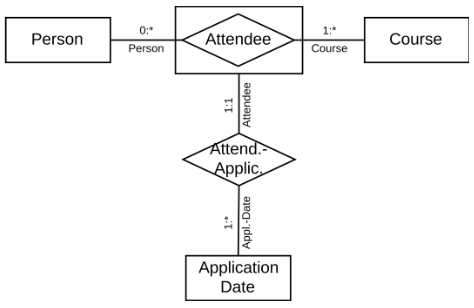


Fig. 7. Objectification

Syntactically, it is important to note that the line of the inner association type must not touch or cross the line of the objectification rectangle. This is important to have a clear distinction where an edge leads to in this combined structure and prevent mistakes when modeling. Like before, objectification is not a newly invented concept, but adopted from existing modeling languages. The origins can be found in the IDEF1X language [10] as well as in existing dialects of ER [23; ch.4.2]

Base and system types

With these rather simple syntax primitives a modeler is already able to depict highly complex situations of the real world. However, the link between this real world model and the technical level is missing at this point. AGILA MOD therefore provides the modeling elements of base types and system types. Both of these types can be perceived as a special form of object type. A system type represents a data type that IT systems use for representing data like e.g. "String", "Integer" or "Boolean". Semantically, it represents all possible data values that a target system can utilize for the particular definition. The concept of a system type is not new. It can be found in other modeling languages like e.g. Step

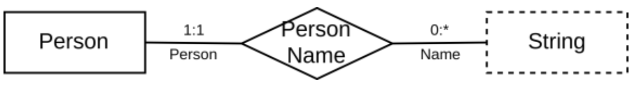
EXPRESS [9] and was adopted into AGILA MOD for offering the connection point to the technical representation.

Additionally, the concept of a base type was introduced. A base type generalizes the concept of a system type to any type of data which by itself has no immediate reference to the real world. A plausible example is the base type “money”. Money is more of a general unit that objects of the real world are applied to. A salary is paid in money for example, but money itself has no meaning whatsoever. Base types can be perceived as an intermediate element of the syntax between an object type and a system type. It is by this best described as a fully formalized lexical object type as it can be found in the NIAM syntax [23; ch.4.5]. As such an intermediate element, it was adopted to AGILA MOD being notated and used the same way as a system type.

In AGILA MOD base and system types are represented as rectangles showing their resemblance to object types, but the border is drawn as a dashed line for easier distinction when reading a model. Graphically, there is no difference between base and system types because they are both used in the same way while modeling. The difference is only of technical nature: while system types have an immediate relation to the data types for implementation, base types need a onetime declaration of how they get mapped to real system types. The advantage of base types is that they regularly have a semantic notion. Furthermore, using a base type several times in a model ensures that the representation keeps unique as the onetime mapping applies for all appearances in the model. Using a system type instead might lead to different representations of the same type of data (like e.g. “money” being represented as numeric value and somewhere else as a float).

Base and system types can be participants of association types like a regular object type. However, when doing so the minimum boundary of this participation basically needs to be “0”.

This becomes obvious when thinking the example in Fig, 8 through: The system type “String” that is used for the association type “Person Name” can analogously to an object type be interpreted as container of all possible strings as arbitrary combinations of characters. Using a lower boundary of “1” would imply that every possible string has to be used at least in one association. This is obviously nonsense and the lower boundary must be “0”.



Fig, 8: System type as part of an association type

Base and system types can also be used to express a representation of a lexical object type. A semantically irreducible sentence for this could be: “A salary is represented by money”. In AGILA MOD, a representation like this is depicted as a directed edge with a dashed line leading from a base or system type to the object type that should be represented.

As a matter of fact, lexical object type always need to have representation as shown in Fig. 9, while non-lexical object types always need to have an identification (like a naming convention or a structuring via an objectified association type) which is also the distinguishing feature between those two kinds of object types.

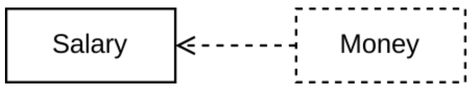


Fig. 9. Base type representing a lexical object type

A “represented by” edge has always its origin in a system or base type and targets to an object type or a base type. The target of a representation must not be represented more than once. With the latter, base types can be represented them by system or other base types to declare their mapping immediately in the model. In some situations, this approach is more useful than leaving the mapping to the derivation. If a base type itself is not represented by a system type, which is explicitly valid, the mapping of a base type to a system type is left to the derivation process where this decision needs to be made once and get documented.

Ancillary elements

Additionally, AGILA MOD introduced two ancillary elements to facilitate the expression of certain real-world conditions in the model that sometimes occur.

An enumeration is a supporting construct that shall explicate the restriction of a basic type to certain values from the beginning, i.e. on the conceptual level. An enumeration acts like a basic type, but its values are restricted to a fixed list of values. In terms of graphical representation, an enumeration is defined as rectangle with a double border on the right side. The border lines are dashed as it is a special form of a basic type. The name of the enumeration is mentioned inside the rectangle. The possible values for the enumeration can be written as textual meta information at the edge of the model. Enumerations should be used in cases where the allowed values for this type are defined permanently from the beginning as a fixed list of unique unordered values (Fig.10).

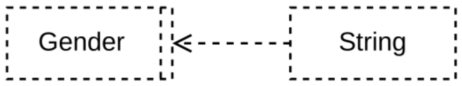


Fig. 10: Enumeration type

As stipulated for the definition of AGILA MOD, the concept of enumeration types is also not new but borrowed in syntax and meaning from the EXPRESS language [22; p. 240]

Similarly borrowed from EXPRESS for AGILA MOD is a set type. It is meant to ease the solution in certain situations that might occur during the modeling phase. A set is an order less collection, free of duplicates, comprising object instances of arbitrary object types. This differs slightly from the set definition in EXPRESS where a set is an order less collection of homogeneous objects.

Graphically, a set is represented as a rectangle with a double border on the left side. Due to the fact that a set is not representing a certain type of object, but is merely a container with arbitrary objects in it, the border lines are dashed. The object types that shall contribute their objects to the set have a directed edge from object type to the set. The set is meant to enable a modeler to associate the containing instances that have no common hierarchy with other object types. This case is pretty seldom. But as Fig. 11 shows, they can occur. In these cases, sets present an easy way out of that problem. The only alternative would be to create an abstract super type from which all participants would have to inherit. This is rather inconvenient and creates an artificially constructed and complex super type structure with no meaning at all that even makes nonsense of the idea of inheritance. One could argument that the set is nothing else then the definition of an artificial super type. However, the meaning is a different one which is an important point for a modeling technique based on semantical formulations. Furthermore, other than a super type, a set does not hand down any properties to the contributing object types.

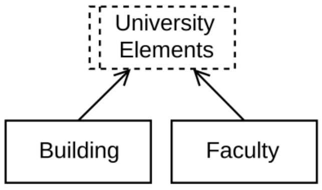


Fig. 11. Set type

Using set types in an association type is subject to certain restrictions. A set type may only be part of an association type with the cardinality of “0:*”. At the same time, the counterpart in this association type may only participate with an upper boundary of “1” (i.e. “x:1” with x ϵ {0, 1). Due to the before mentioned rules concerning association type participants, this restriction implies that a set type can only be part of an association type comprising two participants.

The reason for this restriction lies in the fact that a set type represents merely a lose collection of arbitrary objects. It is only intended to act as a container other object types can refer to. Any other use of this construct results in heavy complications when derivation to logical structures. Fig. 12 shows a vivid example for using a set type for an association type.

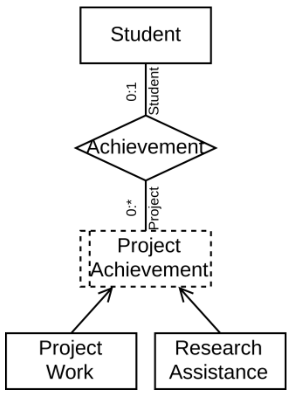


Fig. 12: Using a set type for an association type underlies restrictions

Inheritance

With these syntax primitives at hand, comprehensive conceptual models based on semantically irreducible sentences are possible that also include a bridge to the technical implementation using system types. However, when creating larger models there will be another issue coming up pretty fast. Having a model of a university for example, there will exist students and lecturers. These two object types have something in common in real life: they both represent humans and therefore share quite many facts – both have: first name, surname, birth date, phone numbers, addresses etc. Until now, a modeler would be forced to write down these facts twice (Fig. 13). It is pretty obvious that this is not very efficient and moreover makes a model very soon a lot bigger and confusing. These considerations clearly lead to the concept of inheritance which has its origins in the object-oriented development (cf. [16]).

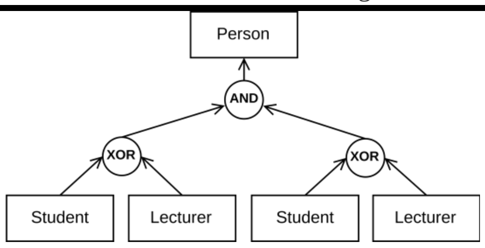


Fig. 13. Utilizing AND in a type hierarchy

Similarly, it has already been introduced into the EER language of Elmasri/Navathe ([1; ch.4]). Syntactically, AGILA MOD follows this concept of EER giving the possibility to express whether the members of the subtypes of a supertype "must be disjoint sets" of "overlapping" ([1; p. 144f.]). For the notation, AGILA MOD uses syntax similar to what is known from UML where a directed solid edge leads from the subtype to the super type. Multiple inheritances with one subtype inheriting from more than one super type are valid. While the concept of multiple inheritance tends to be a problem for programming languages mostly due to behavioral issues of an object, it is controllable in this case as the topic of AGILA MOD is defining data structures, i.e. state and not behavior. Unlike the EER notation, this notation is already familiar for IT experts being used to UML and also comprehensible pretty fast by business experts.

To add the EER concept of disjoint and overlapping subtypes, AGILA MOD was amended by a circle notation set in between the super type and its subtypes which carries a boolean operator. "OR" refers to an overlapping, "XOR" to a disjointed set. This notation also adheres to the idea of semantically irreducible sentences: "A person is either a student or a lecturer" resembles directly an "XOR" inheritance. AGILA MOD also allows the use of the "AND" operator in the circle which can be helpful for more complicated hierarchies like in Fig. 13. It is possible to omit the circle and connect the subtypes directly with the super type. This notation is equivalent with the circle notation using an "OR" operator (Fig. 14; Fig. 16).

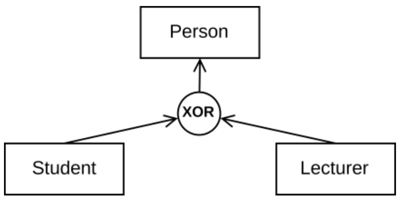


Fig. 14. Subtypes inheriting from a supertype

AGILA MOD also covers the topic which is known both in object-orientation and EER concerning the question if a super type must always be specialized to at least one subtype or not. In [1; ch.4.3, p.145], EER uses the "completeness" constraint for this, where a super type can be totally or only partially member of a subtype. In object-orientation an abstract class can be defined which can't be instantiated on its own. As this concept is again familiar to IT experts and easier to understand by business experts, AGILA MOD adopted this concept to meet the originally stated goals. Any object type can be declared abstract. This also applies for structured object types (i.e. objectified association types). An abstract object type is denoted by drawing the rectangle with a thick line (Fig. 15; Fig. 16).

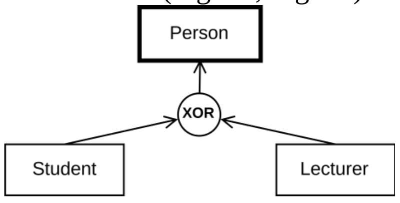


Fig. 15. Using an abstract object type for completeness

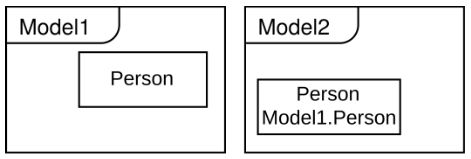


Fig. 16. Object type "Person" of Model1 is used in Model2

Final notes

With these syntax primitives, AGILA MOD enables a modeler to define highly complex conceptual data models creating a truly detailed model of the real world. Finally, two points need to be considered when using AGILA MOD:

– All names of object types and association types need to be unique. Otherwise, it becomes unclear if both object types shall represent the same real-world element or different ones. This does not apply for system and base types as these clearly represent the same thing (A string will always be a string and money will always be money).

– As it is common in software engineering disciplines, elements of a model may be re-used in other models. To be able to reference this element uniquely, the model itself needs to receive a unique identifier for referenciation reasons (A possibility would be the use of an URI).

Approbation of the research material

AGILA MOD has been part of the curricula for computer science and business informatics studies at the faculty of computer science at AUAS for over 15 years. The lessons contain teaching the idea of semantically irreducible conceptual modeling, introducing the syntax of AGILA MOD and accomplishing a self-imposed project task as part of the practical training. The experiences made here are consistently positive. As intended by the mentioned goals, the students are able to understand the philosophy and the basics of this modeling technique pretty fast even if they had no prior experience in this field of activity. This has also been acknowledged regularly in evaluations and feedback talks with students visiting the lectures and tutors that support the students during their practical training. The results in the examinations additionally prove this observation: The number of students failing is to a great extent very low.

As a matter of fact, the courses have been put very early in the curriculum nowadays, i.e. to the second semester for business informatics and the third semester for computer science studies. The faculty council decided to do so after other professors acknowledged that this approach enables students to think through data structures thoroughly and systematically. They can build their teaching upon that basis and facilitates their work.

Furthermore, AGILA MOD has been utilized successfully in several theses of diploma, bachelor and master students. Currently, there is another master thesis in progress using AGILA MOD for creating a conceptual data model acting as basis to derive relational structures from it. AGILA MOD has also been utilized successfully for several projects. The experiences here working on the models together with business experts were as positive as the experiences made with the students during lectures.

Hence, AGILA MOD proves to be eligible to be a modeling language that meets the originally stated goals.

Conclusions and prospects for further research

The presented modeling language AGILA MOD is a good approach to reach the goals formulated at the beginning of this article. It has proven itself in many years of teaching at the faculty of computer science at AUAS and in productive use for several projects and thesis'.

The idea of phrasing semantically irreducible sentences all concerned parties can agree upon which then can be depicted as a simple conceptual model is amazingly simple and easy to grasp for everyone.

By using only a minimum of syntax primitives, the learning efforts that may be necessary to create and read models is minimized. Especially, the concepts described at the end of the main part are only optional and not necessary to start right away. Learning efforts are further minimized by using syntax elements IT experts are familiar with as they are not newly invented but adopted from widely known and accepted modeling techniques (i.e. basically Entity-Relationship Modeling and UML). Only small adaptations and amendments have been added to enable a more precise distinction of semantic meaning or help solving some seldom and complex situations (e.g. base and system types as a special form of object types or sets as a simplification).

By adding the notion of representing semantically irreducible sentences representing the world itself as it is an AGILA MOD model can be understandable for all participants: IT experts (not only database specialists) and business experts. Therefore, AGILA MOD can act as a common platform with IT experts creating the model with their experience and enabling business experts to read the syntax backwards into semantically irreducible sentences and check these against their experience and perceptions.

Furthermore, the idea of semantically irreducible sentence modeling is representing the world or a part of it as “Universe of Discourse”. This leads to the fact that this modelling language is no longer bound to any implementation conditions. It is truly paradigm agnostic and can be derived to any chosen logical data model by applying the according derivation rules.

Further research on AGILA MOD will now be done on the question how the language is already applicable for the newer logical data structures coming up with NoSQL. Hereby, amendments which may be necessary shall be kept minimal to keep the number of syntax primitives still as low as possible. Moreover, there will be research concerning the derivation to logical and physical models. Currently, the ruleset for deriving to relational structures is complete. Hence, rulesets for the other current logical models (like document or graph models) need to be defined. In parallel to this, a development will be started to automate the derivation process to the maximum possible extent.

The goal of this future research is to get a comprehensive framework that receives a conceptual model and produces a logical and physical model by applying the particular ruleset more or less automatically.

References

1. Elmasri, R. & Navathe, S. (2016). “Fundamentals of Database Systems”, Harlow: Pearson Education, 1172 p.

2. Kern-Bausch, L & Jeckle, M. (2001). „Datenbanken“, ch.14 in Schneider U. et al. (eds.). “Taschenbuch der Informatik”, München: *Fachbuchverlag Leipzig*, Germany, 2001, pp.80-88.

3. (2015). Kudraß T. et al. (eds.). „Taschenbuch Daten-banken“, München: *Fachbuchverlag Leipzig*, Germany, 577 p.

4. Kecher, C. & Salvanos, A. (2015). „UML 2.5 - Das umfassende Handbuch“, Bonn: *Rheinwerk Verlag*.

5. Chen, P. (1976). “The Entity-Relationship Model – Toward a Unied View of Data”, *ACM Transactions on Database Systems*, 1976, 1.1, pp. 9-36. DOI>10.1145/320434.320440.

6. Jeckle, M. (2004). „Ableitung Objektorientierter Strukturen aus Konzeptuellen Schemata“, PhD. thesis. Augsburg University of Applied Sciences, Germany, Ulm University, Germany, (May 2004), 375 p.

7. Kastens, U. & Kleine Büning, H. (2018). „Modellierung - Grundlagen und Formale Methoden“, München: *Carl Hanser Verlag*,

8. (1992). Batini, C. et al. “Conceptual Database Design – An Entity-Relationship Approach”, Redwood City: *The Benjamin/Cummings Publishing Company*, 492 p. DOI.org/10.1016/S0065-2458(08)60593-8.

9. ISO/IEC 10303-11: 2004: “Industrial Automation Systems and Integration – Product Data Representation and Exchange”, Part 11: Description Methods: *The EXPRESS Language Reference Manual*, International Organization for Standardization, Geneva, Switzerland and Internet source https://www.sis.se/api/document/preview/905433/.

10. ISO/IEC 31320-2:2012: “Information Technology – Modeling Languages”, Part 2: “Syntax and Semantics for IDEF1X97 (IDEFOBJECT)”, International Organization for Standardization, Geneva: Switzerland and Internet source https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:31320:-2:ed-1:v1:en.

11. Klas, W. & Schrefl, M. (1995).”Semantic data Modeling*”*, Berlin: *Publ. Springer Verlag*.

12. Becket, D. (2004). “RDF/XML Syntax Specification (Revised”). World Wide Web Consortium and Internet source https://www.w3.org/TR/2003/WD-rdf-syntax-grammar-20030905/.

13. Dean, M., & Schreiber, G. (2004). “OWL Web Ontology Language”, World Wide Web Consortium, 2004. Antoniou G., van Harmelen F. (2004). “Web Ontology Language: OWL”. In: Staab S., Studer R. (eds) Handbook on Ontologies. International Handbooks on Information Systems. *Publ. Springer*, Berlin: Heidelberg. DOI 10.1007/978-3-540-24750-0_4.

14. Van Renssen, A. (2003). “Gellish: an Information Representation Language, Knowledge base and Ontology”, ESSDERC 2003, Proceedings of the 33rd European Solid-State Device Research, ESSDERC '03 (IEEE Cat. No. 03EX704), IEEE, 2003 and Internet source https://www.academia.edu/7201584/Gellish_an_information_representation_language_knowledge_base_and_ontology?auto=download.

15. Stellman, A. (2015). “Learning Agile – Beijing: O'Reilly”, Internet source https://www.oreilly.com/library/view/learning-agile/9781449363819/#toc-start.

16. Meyer, B. (1997). “Object Oriented Software Construction”, Upper Saddle River, NJ: *Prentice Hall*, 1225 p.

17. Hull, R. & King, R. (1987). “Semantic Data Modeling: Survey, Application and Research Issues*”, ACM Computing Surveys*, 19.3, pp. 201-260, ACM Computing Surveys, Vol. 19, No. 3, (September 1987), doi>10.1145/45072.45073.

18. Kern-Bausch, L. & Wenzel, B. (1980). „Informationsanalyse. Technical Report“, München: Germany, *Leibnitz Rechenzentrum*.

19. Kern-Bausch, L. & Wenzel, B. (1986). “Database Design for Relational Systems: Why, Who, How?”. European ORACLE Users Group Newsletter, No. 10, doi.org/10.1007/978-3-642-48673-9_3.

20. Kern-Bausch, L. & Jeckle, M. (1998). “From a Semantically Irreducible Formulated Conceptual Schema to an UML Model”, Kern-Bausch, L., Jeckle, M. (1998).” “From a Semantically Irreducible Formulated Conceptual Schema to an UML Model”. In: Schader, M., Korthaus, A. (eds) The Unified Modeling Language. *Physica-Verlag*, pp. 32-44, Part of the book.

21. (1997) .Schader, M. & Korthaus, A. (eds.) „The Unified Modeling Language – Technical Aspects and Applications”, Würzburg, Wien: *Physica-Verlag*, 281 p.

22. Schenck, D. & Wilson, P. (1994). “Information Modeling the EXPRESS Way”, New York: Oxford: *Oxford University press*, 388 p.

23. Loos, P. (1992). „Datenstrukturierung in der Fertigung“, München, Wien: R., *Oldenburg Verlag*, 219 p.

24. Glava, M. G., Malakhov, E. V., Arsiri, O. O. & Trofymov, B. F. (2019). “Information Technology for Combining the Relational Heterogeneous Databases using Integration Models of Different Subject Domains?”, *Applied Aspects of Information Technology*, Vol. 2 No. 1, pp.29-44. DOI://10.15276/aait.02.2019.3.

Received 05.09.2019
Received after revision 15.10.2019
Accepted 30.11.2019

УДК 004.652

[1]**Мюленбах, Сабине**, Ph.D., професор факультету комп’ютерних наук,
E-mail: sabine.muellenbach@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-0392-0334
[1]**Керн-Бауш, Лоре**, Ph.D., професор факультету комп’ютерних наук,
E-mail: lore.kern-bausch@hs-augsburg.de, ORCID https://orcid.org/0000-0003-3401-1333
[1]**Колонко, Маттиас**, Диплом-Wirtschaftsinformatiker (FH), асистент факультету комп’ютерних наук,
E-mail: matthias.kolonko@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-8296-1758
[1]Університет прикладних наук, An der Hochschule, 1, м. Аугсбург, Німеччина, 86161
Tel. +49 (821)55860

МОВА КОНЦЕПТУЛЬНОГО МОДЕЛЮВАННЯ AGILA MOD

Анотація. *У статті розглядаються існуючі підходи до розробки моделей предметних областей з метою з'ясування причин їх слабкого практичного застосування. Визначено важливі вимоги, яким повинна відповідати мова концептуального моделювання для її більш широкого практичного застосування. Крім того, розглядаються концепції мов семантичного моделювання. Використання семантики замість простих структурних конструкцій спрощує доступ і розуміння зацікавлених сторін, які не пов'язані з ІТ. Це допомагає перевірити валідність створених структур даних на відповідність вимогам реального бізнесу. Надалі буде обговорюватися концепція семантично неприводимого моделювання речень, яка може служити мостом між семантичним і концептуальним моделюванням.За результатами цих обговорень представлена мова концептуального моделювання AGILA MOD. Ця мова моделювання заснована на ідеї зображення семантично неприводимих речень в якості графічної моделі. Таким чином, AGILA MOD може виступати в якості загальної платформи, з якої всі учасники проекту можуть домовитися про створення моста між впровадженням ІТ та бізнес-вимогами. Моделі можуть бути створені з семантично неприводимих речень, і їх можна читати назад в семантично неприводимі речення, що робить цю мову легкою для розуміння усіма учасниками проекту. Мова AGILA MOD заснована на відомій мові Entity-Relationship з введенням деяких спрощень. Додано декілька додаткових конструкцій, які також відносяться до добре відомих методів моделювання, що зводить зусилля до вивчення нових елементів майже до нуля. Деривація моделей AGILA MOD в логічну модель виконується за простими правилами деривації, що робить її менш трудомістким і, отже, менш витратним. Ця мова має бути основою для подальших досліджень, спрямованих на нові логічні моделі NoSQL, а також на створення узагальненої структури, яка дозволить максимально автоматизувати процедуру деривації . Крім того, можливість використання концепції багатоваріантної персистентності в поєднанні з AGILA MOD і створення зручного API повинні бути розглянуті в майбутніх дослідженнях*

Ключові слова: *бази даних; концептуальне моделювання предметних областей; семантичне моделювання даних; модель сутність-зв'язок*

УДК 004.652

[1]**Мюлленбах**, **Сабине** Ph.D., профессор факультета компьютерних наук,
E-mail: sabine.muellenbach@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-0392-0334
[1]**Керн-Бауш, Лоре**, Ph.D., профессор факультета компьютерних наук,
E-mail: lore.kern-bausch@hs-augsburg.de, ORCID https://orcid.org/0000-0003-3401-1333
[1]**Колонко, Маттиас**, Диплом-Wirtschaftsinformatiker (FH), ассистент факультета компьютерних наук,
E-mail: matthias.kolonko@hs-augsburg.de, ORCID: https://orcid.org/0000-0002-8296-1758
[1]Университет прикладних наук An der Hochschule 1, г. Аугсбург, Германия, 86161 Tel. +49 (821)55860

ЯЗЫК КОНЦЕПТУЛЬНОГО МОДЕЛИРОВАНИЯ AGILA MOD

Аннотация. *В этой статье рассматриваются существующие подходы к разработке моделей предметных областей с целью выяснения причин их слабого практического применения. Определены важные требования, которым должен соответствовать язык концептуального моделирования для практического применения. Кроме того, рассматриваются концепции языков семантического моделирования. Использование семантики вместо простых структурных обсуждений упрощает доступ и понимание заинтересованных сторон, не связанных с ИТ. Это помогает проверить валидность созданных структур данных на соответствие требованиям реального бизнеса. В дальнейшем будет обсуждаться концепция семантически неприводимого моделирования предложений, которая может служить мостом между семантическим и концептуальным моделированием. По результатам этих обсуждений представлен концептуальный язык моделирования AGILA MOD. Этот язык моделирования основан на идее изображения семантически неприводимых предложений в качестве графической модели. Таким образом, он может выступать в качестве общей платформы, с которой все участники проекта могут договориться о создании моста между внедрением ИТ и бизнес-требованиями. Модели могут быть созданы из семантически неприводимых предложений, и их можно читать обратно в семантически неприводимые предложения, что делает этот язык легким для понимания всеми участниками проекта. Язык AGILA MOD основан на известном языке Entity-Relationship с введением некоторых упрощений. Добавлено несколько дополнительных конструкций, которые также относятся к хорошо известным методам моделирования, сводящим усилия к изучению новых элементов почти до нуля. Вывод моделей AGILA MOD в логическую модель выполняется по простым правилам деривации, что делает его менее трудоемким и, следовательно, менее затратным. Этот язык должен служить основой для дальнейших исследований, направленных на новые логические модели NoSQL, а также на создание всеобъемлющей структуры, максимально автоматизирующей вывод. Кроме того, возможность использования концепции многовариантной персистентности в сочетании с AGILA MOD и создание удобного API должны быть рассмотрены в будущих исследованиях*

Ключевые слова: *базы данных; концептуальное моделирование предметных областей; семантическое моделирование данных; модель сущность-связь*

Müllenbach, Sabine Ph.D., Professor
Research field: Information systems design, database development, conceptual modeling of subject areas, semantic data modeling

Kern-Bausch, Lore, Ph.D., Professor
Research field: Database development, conceptual, logical and physical data models

Kolonko, Matthias, assistant
Research field: Database development for applied information systems, conceptual modeling of subject areas, semantic data modeling