Information Modeling: Forty Years of Friendship

Stig Berild[1] and Eva Lindencrona[2]

[1] Santa Anna IT Research Institute AB, 581 83 Linköping, Sweden
stig.berild@gmail.com
[2] Vinnova, 101 58 Stockholm, Sweden
eva.lindencrona@vinnova.se

Abstract. Information systems design was the theme for a research group at Stockholm University under the leadership of Professor Janis Bubenko, Jr. The research group was called Computer Aided Design of Information Systems (CADIS). The CADIS group developed modeling methods and, at the same time, computer support for the modeling of information systems. The methods were stepwise defined based on early practical applications. Formal modeling languages were defined. The computer support tools turned out to be powerful and grew into a full Database Management System (DBMS), which was further developed into a commercial product (CS5). The CS5 DBMS was based on binary relations. The CADIS approach to modeling and to the DBMS was too early to be commercially successful; however, the basic ideas behind the modeling and the DBMS have survived several generations of "new" approaches to modeling as well as to the DBMS.

Keywords: Binary relations, CASE tools, conceptual modeling, CS5, DBMS, information modeling

1 Introduction

Forty years ago a group of enthusiasts started a research project at the Royal Institute of Technology, Stockholm, under the leadership of Professor Janis Bubenko, Jr. The aim of the research was to develop a new, computer-aided methodology for information systems design. The research project was named Computer Aided Design of Information Systems (CADIS). The methodology aimed to be more flexible, user oriented, and more efficient than the earlier "water-fall" kind of methodologies. Furthermore, what was new by then, the methodology should be computer supported. The computer support – the tool – should be capable of storing the specifications of an information system during its different stages of refinement. These types of tools were later called Computer-Aided Software Engineering (CASE) tools. For the methodology, it was recognized at a very early stage that there was a need for specifications expressed as information and data structures. In addition, it was discovered early on that such specifications needed a CASE tool capable of storing complex structures. A database solution was necessary. Well-defined information and data structuring languages were also needed. This required a powerful but at the same time easily managed DBMS. No such DBMS was available at that time. It was time to step on the information modeling bandwagon.

2 The 1970s

2.1 General Overview

Binary modeling languages had their main period in the 1970s. Langefors [1] was a pioneer with the "elementary message" concept; Bubenko, Jr. [2], Abrial [3], Senko [4] and many others were involved. With Chen [5] came an interest shift towards different types of Entity-Relationship (ER) Modeling Languages. A number of different types of ER modeling languages were defined, mainly as part of some academic research projects. Each language obviously had to include some unique construct – useful or not.

Starting another thread, Codd 1970 wrote [6]. The Relational Model (RM) was born as well as the third normal form. Soon a number of normal forms were defined in research environments. They had their short time "in fame" and then faded away. Usefulness did not seem to be an interesting aspect.

ER-model advocates and RM advocates had their disputes. ER models were mainly aimed at designing conceptual models based on user requirements during the design phase of a database application. Relational schemas were supposed to express the logical structure within the database. Furthermore, the graphical language could hardly be trusted, as it lacked formal definition, or at least an operational language to manage the models.

In 1975, the ANSI/X3/SPARC Study group on database management systems [7] presented its report that introduced a context and a three-layer architecture for database systems. The context was called the "universe of discourse" and the three layers were the external, the conceptual and the internal models. The ANSI/SPARC architectures clarified the need of different models for different purposes, and its terminology and definitions became "de facto" standards among researchers and database developers.

2.2 For Us

Work on methodology for information systems design continued within the CADIS group [8]. A number of different modeling approaches were presented during these years. We used the ANSI/SPARC architecture to compare and identify the usability of different modeling approaches. An analysis of different modeling approaches was presented early in a PhD dissertation [9].

Inspired by the work of Langefors and Bubenko, Jr. as well as by binary languages in general, a CASE tool was developed and implemented. The CASE tool was in itself a DBMS where data elements were expressed and stored as triplets or basic statements (<subject>, <predicate>, <object>). The use of triplets is certainly not unique today – but it was at the time. It was soon discovered that this DBMS could not only easily be used for storing design specifications, but also as a general DBMS. Such a DBMS based on triplets turned out to be extremely flexible and efficient.

During the following years, information systems design became primarily database design and the research on methodology came to focus on formal specification

languages. At the same time, the research and development of a CASE tool turned out to be focused on a general DBMS system including features such as query language, integrated programming language, multi-user environment, transaction management, and so forth rather than emphasizing computer aided solutions. Results were presented at international conferences such as [10] and [11], as well as in a published book [12].

The different versions of the DBMS were called Cadis System 1-5 or the shorter CS1-CS5. The design was a binary model, the schema and implementation was the same binary model. In addition, the integrated programming language was using triplets for storage and retrieval. It represented ease and flexibility in a nutshell. This nutshell also made way for a new approach in systems design; prototype development or experimental programming, as it also came to be known. They could design and implement information systems with full functionality in fractions of the time usually needed using conventional methods. This in turn made it possible to test things, try different functionalities, get a feeling for different user interfaces, which gave the real users a possibility to be directly involved in the system development.

3 The 1980s

3.1 General Overview

The interest for CASE tools and information modeling did not take off as anticipated. Some were using these models merely for conceptual modeling, focusing on the concepts used in an organization, rather than as an input to schema design and implementation. Others really used, for example, ER modeling as part of the design of a database application. However, the bridge from an information model to an RM model was not that easy when it came to real and complex models. The automatic generation of RM schemas was tried, but implementations often needed adjustments from third normal forms down to second normal forms and the like, for performance and other reasons. Keeping the information model and the bridge accurate and updated during the life of the application was not in sight. Modifications were made directly in the RM schema, no wonder CASE tools and information modeling did not take off. Well, to be fair, they did in the Nordic countries. Notwithstanding, this pushed formal specifications, relational DBMS and schemas to the forefront. The relational approach ruled the 1980s.

3.2 For Us

For the CADIS group, research took two directions. On the methodology and modeling side, focus was now on practical tests and the development of formal modeling languages, while on the DBMS side the prototype was developed into a commercial product.

In 1984, Janis Bubenko, Jr. established the Swedish Institute for Systems Development (SISU). Much of the CADIS research on methodology, modeling, and

formal specification languages was now carried out at SISU. The SISU was open to cooperation between research and industry, which led to the practical testing of modeling as a design method for systems- and database applications. Actually, some of the very first practical modeling tests were carried out by SISU. The first SISU publication – SISU Analysis Nr 1 [13] – was completely devoted to modeling and to reports on its practical use. At the same time, formal specification languages were developed. Languages named C-mol and D-mol were developed and presented in reports and papers [11]. Such languages were improved stepwise based on requirements experienced during the practical tests [14]. In addition, the first Swedish textbook on conceptual modeling was published and used in university courses [15].

On the DBMS side, we stood by our binary models. The DBMS was commercialized under the name of DREAM. A number of applications were implemented using DREAM as the supporting database management system. The binary model approach proved to be useful for simple as well as very complex database applications. Complexity in data structures was not a problem, neither were large volumes of data. We were all excited and proud of having developed such a powerful, yet easy to use, database management system. However, according to the commonly held view at the time, databases should be relational. The fact that we were a small company that represented a product not in line with the commonly held truth of the time did not make things easier for us. Consequently, the development efforts had to be reduced after a while. Market pressure towards mainstream DBMS based on the relational model put the creative part of our DREAM to an end.

4 The 1990s

4.1 General Overview

Object-oriented (OO) programming languages became increasingly popular in the 1990s. They represented an approach where data and behavior were integrated into objects. In fact, OO languages were not new at all, although most of the "OO-evangelists" had probably never heard of Simula. Nevertheless, Java soon became popular. Consequently, everything should be an object, even things stored in databases. The objects were just using the database as a place of rest during their dynamic lives. Database and program logic was naturally integrated. Suddenly, all entities in databases should include behavior, even if the information in the database represented a snapshot view of some reality of interest.

For lack of better ideas, the behavior often took the form of "store," "delete," "check value" of the entities in the database, which did not really relate to the behavior of some entity but to the DBMS working with these entities. So what …? Every well-renowned market research company predicted that object-oriented DBMS would replace Relational DataBase Management System (RDBMS) in a few years. Well, things did not turn out that way – for obvious reasons. Information about some reality and work on that information is different things. This, however, does not mean that object databases are useless. They are excellent for certain types of applications,

for instance, those where the application is a reality in its own right, instead of representing some external Universe of Discourse (UoD).

Object-oriented applications needed support from CASE tools, which consequently had to support some object modeling language. Now things started to become interesting, or rather, funny. A fight between proponents of several different design methods and their modeling languages began. Each method had its front figure or so-called "guru." It seemed as if they had not noticed the ER modeling language discussions a decade earlier. Consequently, they represented nothing new except for the inclusion of a simple description of "behavior" in each object type. Each one of these languages was argued to be somewhat better than all the others. Seen from a distance, this fight was nothing more than a strange and somewhat ridiculous episode in the era of information modeling.

Eventually, Object Management Group (OMG) understood that continuing these fights was counterproductive to the future of modeling. OMG took the initiative to melt the different languages down to just one, a language that was later (1996) standardized as the Unified Modeling Language (UML). With UML came a renewed interest in CASE tools. Furthermore, Extendable Mark-up Language (XML) and the increasing demand for information exchange between systems brought in a slight interest in semantics. A tag was attached to each information element, which had to reflect some meaning and understanding of the information element. While a tag did not replace the need for information modeling, it helped to sow an interest in modeling.

4.2 For Us

From 1990 and onwards, a smaller group continued to support DREAM. Applications continued to be supported and new ones were developed. They still exist and work well even today. About 1990, STANLI, a Swedish Standards Institute organization, drew up an interesting approach for information exchange of geographic oriented information, which should be communicated through a central hub. This hub was meant to be responsible for the delivery of information from one party to another. In this role, the hub was also supposed to have the mandate to manage and specify what types of information should be allowed to be communicated. The specification was meant to be in the form of one or several information models. Not only did this model restrict the allowed information, it also represented a uniform semantic view of this information. The hub was supposed to play the role of a "market place" for geographically oriented information. This approach represented new thinking with its focus on interoperability and – even more interesting – interoperability in combination with conceptual models as "rule setters." In fact, the importance of conceptual models for this purpose is still today not fully understood. The semantics is merely being specified using Document Type Definitions (DOD) – and later XML Schema Definition (XSD). Nevertheless, a DOD or XSD schema is not the same as a conceptual model – they complement each other. The conceptual model gives a neutral view of some UoD of interest for exchanging information, while an XML schema defines the content of a specific type of exchange (message type). Regardless, SISU was asked to specify a modeling language for information models. In 1991, the

first version of the STANLI modeling language was introduced. It was based on all the experience collected over a number of years within SISU. The STANLI language was based on a binary view of information and included a number of facilities for expressing semantics. At the time, it had more expressiveness, by far, than most existing languages. This language came to be used extensively, both in connection with geographic information and for modeling in general. To a large extent, the language came to be used for purely conceptual modeling, focusing on the meaning of concepts rather than on designing structures for relational schemas. The STANLI language is still frequently used in Sweden. Conceptual modeling was at the forefront!

5 The 2000 Decennium

5.1 General Overview

The message type definition capability in XML was initially at a very basic level. This changed with the XSD standard from Worldwide Web Consortium (W3C) in 2001. In many ways, XSD had the same expressive power as ordinary modeling languages, although it was intended to express structure and some message type semantics. Nevertheless, the interest for semantics was on the rise.

With the web came the need to specify information about resources on the web (photos, videos, documents, sound). For some strange reason, this type of information came to be called metadata. Some used "metadata" to refer to information models, while others thought of metadata as data describing data elements in databases. Furthermore, no one seemed to have heard of the ISO standard 10027 "Information Resource Dictionary System framework" where different layers of information are clearly defined.

Anyway, the XSD metadata was believed to be so special that it needed its own modeling language. The first version of Resource Description Framework (RDF) became a recommendation of W3C in 1999. Surprisingly enough, RDF was nothing more than triples, i.e., based on a simple binary model. There is nothing wrong with that; however, why reinvent an existing thirty-year-old language just because the UoD is new? Furthermore, why not give at least some references to old sources? The good thing with RDF was that it helped open up a new phase in modeling and in information management, not just for metadata, but for data in general. Data no longer had to be managed in relational databases. With RDF and later recommendations, W3C helped open the door to binary modeling and revitalizing the interest in binary databases. Binary modeling was "on the table" again.

The RDF proponents soon found that RDF could also be used for other types of information than metadata, in fact, for any types of data. What a surprise! The academic world found all this interesting and started to build new features into RDF. Web Ontology Language (OWL) was born, which was mainly reinventing the wheel again. It had features similar to those in ER-modeling languages and UML (not to forget the STANLI language) but with one major different feature; allowing instance

and type specifications to coexist. With RDF and OWL, a model could not be called an information model or conceptual model; it had to be called ontology.

RDF and OWL have their proponents in academic environments. However, in industry and elsewhere, UML became the dominant modeling language, mainly thanks to the support from OMG. Interestingly enough, it came to be used more and more for ordinary information modeling, i.e., without behavior as an ingredient.

Internet interoperability was on the rise. Then along came Service Oriented Architectures (SOA) with Web Services Architecture as the most well-known alternative. XML and XSD were part of this movement into information exchange. No longer was information just stored in databases, it was moving around on the web. No longer were applications something internal to an organization. They were redesigned as services and open to the global community to use as long as it was benefitting the business.

5.2 For Us

Information of different kinds – be it data, metadata, model data, or anything else – constitutes the base ingredient that gives the web its purpose, meaning and soul. It is about knowledge sharing, cooperation, and information exchange in its most general sense, and it is about information and service interoperability (exposure, discovery, interpretation, exchange, and use).

Lacking in SOA was and is a layer expressing an independent view exchange of some UoD of interest. It was and is believed that XML and XSD are doing the job. However, as it has already been pointed out, message structures using XSD and UoD based information models are not the same, they complement each other.

Furthermore, is XML really the optimal way of delivering information? Not necessarily. Information could just as well be delivered as chunks of triples or elementary messages. Together, these triples express exactly the same thing as a message expressed in XML. Moreover, these chunks may be sent as a file with the same internal format as "our" binary database. The chunk is, in this way, a database in itself and, as such, operable through the usual database interface. Suddenly, not only messages, but also whole databases could exist moving around in cyber space or in a permanent place. Why not mark this change of view by replacing "database" with "information cluster."

Our interest in information modeling has continued and a number of projects have been carried out. One example of such a project was Eurocontrol/OATA (Overall ATM/CNS Target Architecture), a major air traffic control project aiming at improving the overall performance, safety, and sustainability of European air transport. A very large and complex UML model played a major role in the OATA specification. Other examples are projects in the health care sector.

6 Final Thoughts

We started with binary models in the CADIS project, went through all the turbulence during the years, and have just reached the end of the National Information Structure project [16] where STANLI has been used as the modeling language.

DREAM applications are still in use. STANLI is still a valid choice as a modeling language, and will so be in future projects.

Binary modeling combining simplicity and enough expressiveness is an attractive combination. The fact is that we still draw bubbles and arrows on napkins during lunch discussions, prefer to use simple general-purpose graphical tools when they are available, and sometimes work with more advanced tools in formal projects. So much has happened during the years, but most things are still the same.

The described forty years constitute a full loop of information modeling and modeling languages.

References

1. Langefors, B.: Theoretical Analysis of Information Systems. Studentlitteratur, Lund, Sweden (1966, 1973)
2. Bubenko, Jr., J.A.: The Temporal Dimension in Information Modelling. In: Proceedings of IFIP WG 2.6 Working Conference on Architecture and Models in Data Base Management Systems (1977)
3. Abrial, J.R.: Data Semantics. In: Klimbie & Koffeman (eds.) Data Management Systems. North-Holland (1974)
4. In: Morlet & Ribbens (eds.): International Computing Symposium, North-Holland (1977) Senko, M.E.: Conceptual Schema, Abstract Data Structures, Enterprise Descriptions.
5. Chen, P.P.: The Entity-Relationship Model – Toward a Unified View of Data. In: ACM Transactions on Database Systems (1976)
6. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM (1970)
7. ANSI/X3/SPARC Study Group on Database Management Systems, Interim Report (1975)
8. Bubenko, Jr., J.A., Berild, S., Lindencrona-Ohlin, E., Nachmens, S.: Information Analysis and Design of Data Base Schemata. TRITA-IBADB- 3091, Department of Information Processing and Computer Science, Royal Institute of Technology and Stockholm University (1975)
9. Lindencrona-Ohlin, E.: Determination of Information Structures. TRITA-IBADB-0098; Department of Information Processing and Computer Science, Royal Institute of Technology and Stockholm University (1976)
10. Berild, S., Nachmens, S.: CS4: A Tool for Database Design by Infological Simulation. In: Proceedings of Very Large Data Base Conference (VLDB1977), Tokyo (1977)
11. Lindencrona-Ohlin, E.: A Study of Conceptual Data Modelling. PhD dissertation. Chalmers Technical University and University of Gothenburg (1979)
12. Janning, M., Nachmens, S., Berild, S.: CS4 – An Introduction to Associative Data Bases and the CS4 System. Studentlitteratur, Lund (1979)
13. SISU Analys, Nr 85/1: Konceptuell modellering, SISU – Svenska institutet för Systemutveckling (1985)
14. Lindencrona-Ohlin, E., Bubenko, Jr., J.A.: Towards a Formal Syntax for a Data Modelling language –DMOL, SYSLAB Working Paper Nr 63, (1983)

15. Bubenko, Jr., J., Lindencrona, E.: Konceptuell modellering – Informationsanalys, Studentlitteratur, Lund (1984)
16. The National Board for Health and Welfare: Nationell informationsstruktur. www.socialstyrelsen.se/NI