2007

A FRAMEWORK THAT ENABLES THE USE OF EXPERIENCE IN CONCEPTUAL MODELLING

Paulo Tomé
Polytecnic of Viseu, ptome@di.estv.ipv.pt

Ernesto Costa
University of Coimbra, ernesto@dei.uc.pt

Luís Amaral
University of Minho, amaral@dsi.uminho.pt

A FRAMEWORK THAT ENABLES THE USE OF EXPERIENCE IN CONCEPTUAL MODELLING

Tomé, Paulo, Polytecnic of Viseu, Viseu, Portugal, ptome@di.estv.ipv.pt

Costa, Ernesto, University of Coimbra, Coimbra, Portugal, ernesto@dei.uc.pt

Amaral, Luís, University of Minho, Guimarães, Portugal, amaral@dsi.uminho.pt

Better Information Systems (IS) could be developed, if experience were used. The use of experience could be applied in any phase of the Information System Development process. This paper presents a framework that applies the Case-Based Reasoning (CBR) method to enable the use of experience in conceptual modelling. This framework could be used in several conceptual modelling types, as long as graphical modelling languages were used. Our framework was implemented in an Internet application that has a modular structure.

Keywords: Conceptual modelling, Case-Based-Reasoning, Re-Using Experience

1 INTRODUCTION

Conceptual models play an important role in several organizational activities, because they are an important knowledge source for business decisions (Berger and Pfeiffer 2007). It is also important to notice that conceptual models are an important tool in Information Systems Development (ISD) process (Krogstie and Solvberg 1999). According to Krogstie and Solvberg, conceptual models are used to represent system requirements and form a basis for system design and implementation. Besides that, they are a vehicle for communication and could be use for documentation and sense-making.

Generally, in an ISD process IT professionals developed several kinds of conceptual models. Conceptual data models and conceptual functional models are two examples of models developed by IT professionals. Each kind of conceptual model is generally developed using a specific perspective.

The use of experience in conceptual modelling plays an important role. Generally, senior IT professionals develop better models than novice IT professionals (Batra and Davis 1992). Senior IT professionals apply their experience in new situations.

Since the 90s, several authors have done research into the re-using of experience in conceptual modelling (Tauzovich 1990; Lloyd-Williams 1994). But despite this, the re-use of experience in ISD is still an area of research. For example, last year a unified architecture of experience engineering was proposed (Sun and Huo 2006). In 2003 the pattern paradigm was extended to embrace useful functionalities necessary in re-use of experience in ISD (Purao, Storey et al. 2003).

This paper presents the use of a new approach, the CBR method, for re-using experience in conceptual modelling. In section 0, we explain the main aspects of conceptual modelling. The CBR main functionalities are explained in section 0. In section 0, we present our framework that enables the re-use of experience in ISD conceptual modelling and the results obtained with its application in conceptual data modelling tasks. Finally, in section 5, we present ours conclusion and future remarks.

2 CONCEPTUAL MODELLING

Conceptual models are generally constructed during the problem analysis and requirements specification of the ISD process (Krogstie and Solvberg 1999). The conceptual models are an important knowledge repository for future organizational processes. As previously mentioned, generally in ISD processes several types of conceptual models are developed.

There is not a consensual framework for classifying the conceptual models. Generally, one conceptual model type is used to express one system perspective. For each conceptual model type, several notations exist, most of them diagrammatic (also called modelling languages). Krogstie and Solvberg (Krogstie and Solvberg 1999) consider that the currently conceptual models languages enable the definition of the perspectives: structural, functional, behavioural, rule, object communication and actor role.

In the bibliography of the ISD domain, several modelling languages were described. Two of the most commonly used perspectives in ISD process are the data and functional perspectives. The Chen (Chen 1976) ER modelling language is one of the oldest notations used in conceptual data modelling, whilst the DFD notation (Gane and Sarson 1979) is one of the oldest notations used to express the functional perspective.

If a diagrammatic notation is used, the conceptual model is a graph. As shown in Table 1, each modelling tool has constructors that can be used to represent graph node elements and constructors that can be used to represent graph edge elements. Generally, the set of graph elements is different from modelling tool to modelling tool.

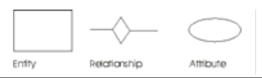
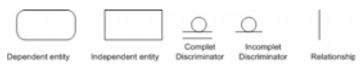
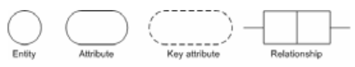
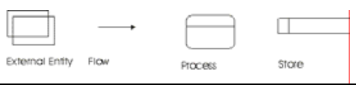
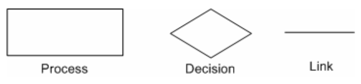
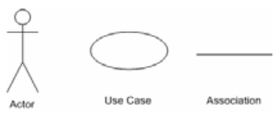
Modelling Languages	**Type**	**Constructors**
ER - Chen Notation (Chen 1976)	Structural	
IEDF1X (FIPS 1993)	Structural	 Dependent entity Independent entity Complet Discriminator Incomplet Discriminator Relationship
NIAM (Halpin and Nijssen 1989)	Structural	
DFD (Gane and Sarson 1979)	Functional	
Flowcharts (Gane and Sarson 1979)	Functional	
Use Cases (Fowler and Scott 1997)	Actor	

Table 1 Some Conceptual Modelling Languages

The set of constructors of the modelling tool and the way they can be used is generally a restriction for the IT professionals. Some of the IT professionals overlap this restriction adapting the modelling tool to their needs.

3 THE CBR METHOD

CBR is a methodology (Watson 1999) that tries to solve new problems based on solutions for similar previous ones (Kolodner 1993; Aamodt and Plaza 1994). CBR is based on two crucial aspects: the *cases* and the resolve process model.

The *case* is formed by the *problem* and the *solution* (Kolodner 1993). The *objective* and the *characteristics* of the situation are described by the *problem*. The *solution* consists of the solution itself, the solution evaluation and reasonings. The identification of *cases* types constitutes the major step forward in the development of the CBR system. The set of *cases* of the CBR system is called *case memory*. An important issue related to *cases* is the indexing, which creates a label associated to the case that will allow us to remember it.

The resolve process, called CBR Cycle, begins with the problem description and ends with the solution. The CBR cycle has two principal models: 4Rs proposed by Aamod and Plaza (Aamodt and Plaza 1994) and the one proposed by Kolodner (Kolodner 1993). The CBR cycles generally involves the following activities:

- case search to find similar cases;
- similarity evaluation to measure the level of similarity between the problem that needs solving and the stored ones;
- adaptation to adjust one or several solutions to the current problem;
- case retain to store the new resolved problem.

The case search is based on the problem description and the similarity evaluation is based on similarity functions (Althoff, Auriol et al. 1995). Consequently, the new solution is built by adapting old solutions to the needs of the current problem. The last task of the CBR cycle is the inclusion of the *case* in cases memory. Given the fact that a new *case* is added to the system, it could be said the CBR systems have the ability to learn.

It is important to mention, that there are a lot of domains where the CBR methodology has been used (Kolodner 1993; Watson 1996; Mántaras and Plaza 1997). For example, CBR has been applied in software development, architectural design, meal planning and legal reasoning systems.

The problem presented in this paper could be classified as belonging to the design class of the classification schema proposed by Althoff (Althoff, Auriol et al. 1995). The developing of conceptual models is a design task because the model conception is carried out without any guidelines. There are several CBR systems that share this property. These are mainly found in the software development environments where it is possible to reuse software code. The Rebuilder project (Gomes 2006) is an example of this and aims to use the CBR methodology in the development of UML diagrams (Gomes, Pereira et al. 2002; Gomes, Pereira et al. 2003; Gomes, Pereira et al. 2003). The Experience Factory (Althoff, Nick et al. 1999) proposes a structure and a software application that aims to reuse experience in the context of software development processes. Krampe and Lusti (Krampe and Lusti 1997) applied CBR in the IS design, but the emphasis of this work was on the use of design specifications. Their focus is also on software development process.

Regarding this work, it is important to say that this framework is not concerned with the software development process (i.e. code writing). It is meant to help the development of conceptual models. However, the use of UML diagrams could be an area of common ground with the Rebuilder project. We consider our framework as a tool that contributes to good Knowledge Management (KM). The KM leads to rational allocation of organisational knowledge assets (Althoff and Weber 2005).

4 THE USE OF EXPERIENCE IN CONCEPTUAL MODELLING

It is generally accepted in IT domain that a system that enables the re-use of experience must search and adapt old solutions (Freeman 1987). Our framework implements these functionalities and others that we think are also important.

A system can be considered successful if, first, it allows several software modelling tools to use it, and, secondly, if it allows several conceptual modelling languages to use it. Besides this, the system must be available through Internet technology.

The framework proposed, shown in Figure 1, has two main parts: the client and the server. The client consists of a browser and a software modelling tool. The server has several components that enable the use of several software modelling tools, the use of several modelling languages and the re-use experience. Several software modelling tools can be used so long as they export data to XML format.

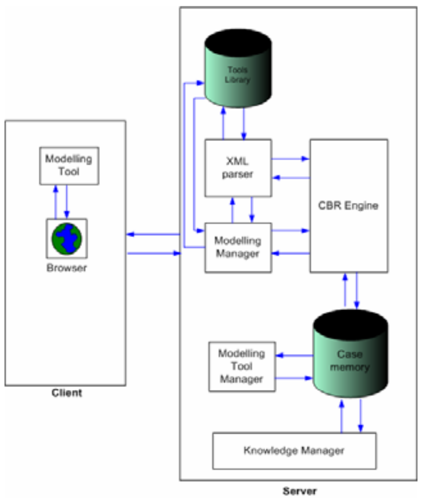


Figure 1 - Framework

The server components that enable the use of several software modelling tools and several modelling languages are: *Tools library*, *XML parser* and *modelling manager*. The *tools library* is a repository of parsing rules that enable parsing files to extract information about stored conceptual models. Besides that, the *tools library* stores information on how to communicate the re-usable models (or part of them) to the browser. The *XML parser* extracts information contained in model files. The *Modelling manager* is responsible for the communication between the server and the client.

The CBR method is implemented through the: *Case memory*, *CBR engine*, *Modelling tool manager* and *Knowledge manager*. The *CBR engine* implements the 4Rs cycle proposed by Aamodt and Plaza (Aamodt and Plaza 1994). The *Modelling tool manager* is responsible for generating knowledge about the modelling languages. The *Case memory* stores past resolved situations and general knowledge domain. Finally, the *knowledge manager* is responsible for managing the set of past resolved situations.

The *Case memory* stores the past resolved situations (cases) and the general knowledge domain. We considered that four types of cases could be useful. First, it could be useful to have complete models. But, besides that, it could also be useful to have individual model constructors. For this last purpose, we consider three constructor types: *node*, *arc* and *attribute*. As mentioned in section 1 the diagrammatic modelling languages have two main constructor types: nodes and arcs. But besides these two types, the constructors of some modelling languages have variable number of elements. This happens, for example, in the entity constructor of the IDEF1X language.

We considered that each case type, a part from its solution, consists of a set of characteristics that describe the structure of the situation and the context of the situation. For the model case type, the number of nodes and the number of edges by node are two characteristics related with the model structure. Although, the keywords of the model nodes is a semantic characteristic that contextualizes the conceptual model.

Each modelling language is described through the meta-case structure, shown in Table 2. But each specified characteristic has a label that specifies its role in this particular constructor.

Case Type	Description		
Model	**Problem** Objective: Model definition Characteristics Organization Keywords Type of organization Type of description Modelling tool Node keywords Number of nodes Number of node links **Solution** XML description	Node	**Problem** Objective: Node definition Characteristics Type of description Node keywords Number of attributes Modelling tool Type of description Attribute keywords Linked with Node characteristics **Solution** XML description
Attribute	**Problem** Objective: Attribute definition Characteristics Type of description Attribute keywords Belongs to Attribute characteristics Modelling tool Type of description Attribute characteristics **Solution** XML description	Arc	**Problem** Objective: Arc definition Characteristics Type of description Arc keywords Arc attributes Modelling tool Type of description Link Arc characteristics **Solution** XML description

Table 2 Meta-Case structure

Besides the past resolved situations, the *case memory* has specific knowledge about each modelling language. The weight of each characteristic and the adaptation rules were also stored.

We apply clustering techniques to structure the *case memory*. Because a problem could be searched by several combinations of its characteristics, we defined a structure that addresses cases by all possible characteristics. Inside each combination we use clustering techniques to group cases into sets. For each case group we define which case is medoid, e.g., the case that best describes the set of cases.

The *CBR engine* follows the 4Rs cycle (Aamodt and Plaza 1994). The Recall, Re-Use, Revision and Retain phases were implemented. The cycle begins with the characteristics' specification. The Recall phase begins after the characteristics' definition (the algorithm is shown in Table 3). We determine the neural network output to obtain the group of clusters where we will find the cases that are most similar to the current problem. The current problem is evaluated regarding to each cluster medoid. This evaluation will enable us to find which cluster will be used to find the cases that are most similar to the current situation.

1. Find the cluster group (based in a neural network)
2. Find the cluster in which to do the similarity evaluation
3. Similarity evaluation

Table 3 Recall Phase Algorithm

The Re-Use phase copies the equal case parts and tries to adapt the different parts. The adaptation is done through adaptation rules defined in the modelling language domain knowledge definition step or by rules generated during the framework usage.

The Revision phase is generally done by the framework user. The solution proposed by the CBR engine is edited by the user and he makes the necessary corrections. Finally, the case is inserted in the cluster groups. Each cluster medoid, where the case was inserted, is recalculated.

Through the *Modelling tool manager* the framework user defines the modelling language knowledge domain. Based on the meta-case structure, the user specifies the constructor characteristics. The role of characteristic for each specific constructor is specified. For example, the node characteristic *linked with* in a data modelling tool has the role *related with*. Besides that, the adaptation rules and the weight of each characteristic are also defined.

The *Knowledge manager* aims to manage the case memory. It could be that some past situations were not well modelled. The framework user, through the *Knowledge manager*, makes these situations unavailable.

The framework was tested with fifteen IS data models. The data models belong to different organizations/domains. Each data model has a different number of entities, attributes and relationship as described in Table 4. We used two different strategies. In the first strategy, the system was launched separately for each data model (no models in memory). In the second strategy, the data models are introduced sequentially from M_1 to M_{15}.

	M_1	*M_2*	*M_3*	*M_4*	*M_5*	*M_6*	*M_7*	*M_8*	*M_9*	*M_{10}*	*M_{11}*	*M_{12}*	*M_{13}*	*M_{14}*	*M_{15}*
Number of entities	*4*	*24*	*17*	*13*	*10*	*6*	*22*	*16*	*7*	*8*	*4*	*4*	*30*	*4*	*4*
Number of Attributes	*6*	*86*	*113*	*44*	*48*	*60*	*102*	*67*	*22*	*74*	*25*	*22*	*130*	*13*	*53*
Number of relationships	*4*	*32*	*12*	*10*	*9*	*4*	*12*	*13*	*3*	*7*	*3*	*3*	*44*	*4*	*3*

Table 4 Number of constructors

When the models are launched without any previous model in memory, there is a low percentage of adapted constructors (results shown in Table 5).

	M_1	*M_2*	*M_3*	*M_4*	*M_5*	*M_6*	*M_7*	*M_8*	*M_9*	*M_{10}*	*M_{11}*	*M_{12}*	*M_{13}*	*M_{14}*	*M_{15}*
Adapted entities	0%	*0%*	*5.9%*	*0%*	*10%*	*0%*	*9.1%*	*6.3%*	*28.6%*	*0%*	*0%*	*0%*	*10%*	*0%*	*0%*
Adapted Attributes	66.7%	*24.4%*	*39.8%*	*52.3%*	*22.9%*	*25%*	*34.3%*	*32.8%*	*45.5%*	*28.4%*	*32%*	*22.7%*	*36.2%*	*23.1%*	*58.5%*
Adapted relationships	75	*96.9*	*91.7*	*90*	*88.9*	*50*	*91.7*	*92.3*	*66.7*	*85.7*	*66.7*	*66.7*	*97.7*	*75*	*66.7*

Table 5 Situation 1 Results

As we can see in Table 6 when the models are launched sequentially the percentage of adapted cases increases significantly. For instance, the case relationship is in almost all situations derived by adapting cases contained in the case memory. By contrast, for entity cases the use of past cases is very low. This can be justified by the heterogeneity of IS domains. Notice also that the order of data models created was not considered an issue.

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
Adapted entities	0%	8.3%	5.9%	7.7%	20%	0%	9.1%	12.5%	28.6%	25%	0%	0%	20%	25%	25%
Adapted Attributes	66.7%	26.7%	48.7%	70.5%	43.8%	45%	41.2%	50.8%	77.3%	43.2%	60%	68.2%	48.5%	92.3%	83%
Adapted relationships	75%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 6 Situation 2 Results

5 CONCLUSION AND FUTURE REMARKS

We have empirically shown that the inclusion of a CBR methodology providing a memory of past experience could greatly improve the conceptual modelling task. The use of adapted cases benefits the user since he/she does not need to provide that information manually to the system. Therefore the ISD could focus on the new elements.

Nonetheless, this project needs some improvements. One improvement must be the development of modelling language converting mechanisms. This kind of mechanism will enable the usage of different modelling languages to model the same conceptual perspective. Another is the need for the development of parsing rules for the most representative software modelling tools.

6 REFERENCES

Aamodt, A. and E. Plaza (1994). “Case-Based Reasoning: Foundational Issues, Methodological Variations and Systems Approaches.” AI-Communications **7**(1): 39-52.

Althoff, K. D., E. Auriol, et al. (1995). A Review of Industrial Case-Based Reasoning Tools, AI Intelligence.

Althoff, K. D., M. Nick, et al. (1999). CBR-PEB: An Application Implementing Reuse Concepts of Experience Factory for the Transfer of CBR System Know-How. 7th German Workshop on Case-Based Reasoning, Wurzburg.

Althoff, K. D. and R. O. Weber (2005). “Knowledge Management in Case-Based Reasoning.” The Knowledge Engineering Review **20**: 305-310.

Batra, D. and J. G. Davis (1992). “Conceptual Data Modelling in Database Design: Similarities and Differences Between Expert and Novice Designers.” International Journal of Man-Machine Studies **37**: 83-101.

Berger, J. and D. Pfeiffer (2007). Automatic Knowledge Retrieval from Conceptual Models. CAISE'07, Trondheim - Noruega, Tapir Academic Press.

Chen, P. P.-S. (1976). “The Entity-Relationship Model - Toward a Unified View of Data.” ACM Transactions on Database Systems **1**(1): 9-36.

FIPS (1993). Integration Definition for Information Modeling (IDEF1X), Federal Information Processing Standards Publications.

Fowler, M. and K. Scott (1997). UML Distilled - Applying the Standard Object Modeling Language, Addison-Wesley.

Freeman, P. (1987). Reusable software Engineering: Concepts and Research Directions. IEEE Tutorial Software Reusability.

Gane, C. and T. Sarson (1979). Structured Systems Analysis: Tools and Techniques, Prentice-Hall.

Gomes, P. (2006). Rebuilder. rebuilder.dei.uc.pt, Paulo Gomes.

Gomes, P., F. C. Pereira, et al. (2002). Using WordNet for Case-Based Retrieval of UML Models. STarting Artificial Intelligence Researchers Symposium (STAIRS'02).

Gomes, P., F. C. Pereira, et al. (2003). Case-Based Reuse of UML Diagrams. Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'03).

Gomes, P., F. C. Pereira, et al. (2003). Human-Machine Interaction in a CASE environment. International Joint Conference on Artificial Intelligence IJCAI'03 Workshop: "Mixed-Initiative Intelligent Systems".

Halpin, T. A. and G. M. Nijssen (1989). Conceptual Schema and Relational Database Design, Prentice-Hall.

Kolodner, J. (1993). Case-Based Reasoning, Morgan Kaufmann Publishers.

Krampe, D. and M. Lusti (1997). Case-Based Reasoning for Information Systems Design. ICCBR-97.

Krogstie, J. and A. Solvberg (1999). Information Systems Engineering: Conceptual Modeling in a Quality Perspective. Trondheim, The Norwegian University of Science and Technology.

Lloyd-Williams, M. (1994). “Expert System Support for Object-Oriented Database Design.” International Journal of Applied Expert Systems **1**(3).

Mántaras, R. L. and E. Plaza (1997). “Case-Based Reasoning: An Overview.” AI Comunication **10**(1): 21-29.

Purao, S., V. C. Storey, et al. (2003). “Improving Analysis Pattern Reuse in Conceptual Design: Augmenting Automated Processes with Supervised Learning.” Information Systems Research **14**(3): 269-290.

Sun, Z. and H. Huo (2006). The Engineering of Experience. Sixth International Conference on Intelligent Systems Design and Applications.

Tauzovich, B. (1990). An Expert System for Conceptual Data Modelling. 8th International Conference on the Entity-Relationship Approach, Toronto, Canada.

Watson, I. (1996). Case-Based Reasoning Tools: an Review. 2nd UK Workshop on Case-Based Reasoning, University of Salford, AI-CBR/SGES Publications.

Watson, I. (1999). “CBR is a Methodology not a Technology.” Knowledge Based Systems Journal **12**(5-6).