The Object Relational impedance mismatch from a cognitive point of view

Laura Benvenuti
Hogeschool van Amsterdam
Wibautstraat 2-4
1091 GM Amsterdam – The Netherlands
lbe@acm.org

Gerrit C. van der Veer
Sino European Usability Center,
Dalian Maritime University,
Dalian, 116026 China
gerrit@acm.org

Keywords: POP-I.A. group characteristics, POP-II.C. working practices, POP-V.A. mental models

Abstract

Computer Science has evolved towards a discipline with different branches. Scholars study and define artefacts from different viewpoints: computer languages, environments, paradigms. Practitioners work in multidisciplinary teams. They design, produce and link software that was designed according to different paradigms. We are interested in the communication between these practitioners. Do they refer to the same concepts when they use the same words? We designed an experiment to assess this.

1. Introduction

Computer Science has developed different branches, each with its own body of knowledge and its own problem area. Object Oriented software and relational databases have their origins in different approaches to the digitalization of information. There are evident differences between the Object Oriented (OO) and the Relational paradigm. The concepts underpinning the relational model are prescriptive and formal, while those underpinning object schemas are more descriptive (Ireland,2011).

What happens when the paradigms meet, when engineers link together software that was designed from different points of view? This area is characterized by a persistent problem, the Object Relational impedance mismatch. Many attempts have been made to solve the problem by developing new software (Object Relational Mappings), where tables correspond to classes and rows correspond to objects. This is problematic, though. In an OO program, an object has an identity independent of its state, but specific to the program execution. In the relational model, the row is identified by its attribute values (the state of the row) and it is accessible through set operations only: not directly. Direct access to objects during the execution of OO programs is possible and is based on navigation.

The problem with these solutions is their validation. Computer scientists, in particular database engineers, work on abstract entities. Technical solutions of the Object Relational impedance mismatch can only be evaluated if working practitioners perceive a common field of application, if they work with the same abstract entities.

In this paper, we focus on mental representations of abstract entities. In section 2 we explore the notion of a mental model. In section 3 we give an overview of research on cognitive aspects of programming and database interaction. Reflection follows in section 4. In section 5 we describe an experiment to assess differences in mental models between programmers and database professionals. The results until now are shown in section 6; preliminary conclusions are drawn in section 7.

2. Backgrounds

2.1. Mental models

According Craik, that the mind constructs “small-scale models” of reality and uses them to anticipate events (Johnson-Laird, 1989). Johnson-Laird formulated a theory of mental models, meant to explain human thinking and reasoning. Norman (1983) is more specifically concerned with mental models in Human-Computer Interaction. Users, states Norman, construct mental models of computer systems incrementally, while interacting with systems. The resulting models are constrained by the users’ prior knowledge, needs and context. These models are often incomplete. They are limited by the human

information processing system, by experience and by needs that can be contrasting: the need to focus and the need to retain important details. They are parsimonious in order to reduce mental complexity. User mental models are unscientific and unstable. They evolve over time: people learn, people forget. People use metaphors to simplify their models. Nevertheless, mental models are functional to support tasks such as planning, execution, assessment of results and understanding of unexpected events.

Despite the attention to mental models and their conception, there is little agreement on the exact definition of the term "mental model". Does the term refer to temporary structures in Working Memory (WM) or knowledge structures in Long Term Memory (LTM)? Cañas and Antolí (1998) introduce this definition: a mental model is "*the dynamic representation that is formed in WM combining the information stored in LTM and the extracted information from the environment*". From now on, we will follow Cañas' and Antolí's definition.

Human WM is limited. People take these limitations into account and adopt strategies to keep mental models manageable in WM. The question here is: can we assume mental models of practitioners with different backgrounds to be compatible with each other?

2.2. Assessing mental models: the teach-back protocol

Mental models can not be observed directly. Van der Veer (1990) extended an hermeneutic method designed by Pask, intended to elicit information about mental models (the Teach-Back method), and adopted it to detect differences in mental representations of users interacting with computers.

A situation is simulated where the respondent has to interact with a computer. The participant is asked to explain the computer's functioning to an imaginary counterpart, a colleague or a student, who has similar experience with the situation. The questions are designed to activate both declarative and procedural knowledge structures. They are presented on white sheets of paper, and the participants are instructed to express themselves in whatever way they consider most adequate: text, drawings, keywords, diagrams etc. In this manner, the participants are encouraged to externalize the mental model they made of the situation. Next, the protocols are scored "blind", along pre-defined scoring categories in order to map the respondent's mental representations. Rating implies (1) reading the protocol in its entirety and trying to understand fully what it says; (2) trying to formulate how the participant understands the space of the teach-back question and (3) classify the responses into relevant categories for the purpose of the study. Rating the answers is a complex task: the rater has to interpret the participant's intention and classify it by means of scoring rules. This task requires considerable training. In order to safeguard reliability, scoring is done independently by two or more persons.

3. Literature review

In the next sections, we will review the literature on understanding of programming constructs.

3.1. Cognitive aspects of (OO) programming

Robins, Rountree and Rountree (2003) provide us with an extensive literature review on research concerning learning and teaching programming between 1970-2003. In a survey study, Détienne (1997) assesses claims about the cognitive benefits of the OO paradigm.

Robins et al. mention different kinds of mental models involved in learning programming. Many studies have noted a central role played by a model (or abstraction) of the computer. Du Boulay et al. (1989) call this the "notional machine", an idealized, conceptual computer that is defined with respect to the language. Novices should develop an appropriate notional machine to master a programming language: the notional machine underlying Pascal is very different from the machine underlying PROLOG. A study by Mayer (as cited by Robins et al, 2003) confirms that students supplied with a notional machine model perform better than students who are not given the model.

A model of program comprehension is provided by Pennington (1987). Comprehension occurs in the context of a problem domain. The program's text is re-organized in mental representations with help of available knowledge structures. Pennington studies expert FORTRAN and COBOL programmers, and finds significant differences in their performances of comprehension tasks.

The OO approach has claimed to make modeling the problem domain easier for programmers. Détienne investigates this claim. She finds that novice programmers have difficulties in class creation. They need to start with a procedural representation of the situation. This seems to indicate that knowledge is organized in terms of procedures, not in terms of objects and relations. But for expert OO designers, the claims find support. Expert OO designers seem to shift between object view and procedure view.

3.2. Cognitive aspects of user-database interaction

Cognitive aspects of query languages were studied at the same time as cognitive aspects of programming (Reisner, 1981). One of the issues was the procedurality of the query language. Some query languages specify more step-by-step methods to obtain results than others. SQL, today's standard, is a set-oriented language, and was been labeled "non-procedural" in the debate.

There is no need to know how data are stored in a Relational Database in order to write a query: knowledge of the (abstract) data model of the database is sufficient. Chan, Wei and Siau (1993) focus on cognitive processes of abstraction in the user-database interface. They distinguish three levels of abstraction: a conceptual level (a description of the user's world), a logical level (describing the database world in mathematical terms) and a physical level (describing states in computer memory). De Haan and Koppelaars (2007) explicitly address database professionals. Professionals need to control the RDBMS, which is why database engineers should master the logical level, and the database world's description in terms of set theory. Database professionals work with "objects" in the database world, a mathematical construct. No studies on the cognitive aspects of this kind of user-database interaction are known to us.

4. A need for empirical study

The question here is how programmers, designers and professionals characterize the abstract software structures they work with. We observe that this characterization can change over time. Novice OO programmers mainly adopt a procedural strategy, expert OO designers are able to switch between object and procedure view. Knowledge of RDBMS software is not needed to formulate queries, but expert database professionals often do understand what is " inside the machine". Database experts, too, switch between abstract (relational) view and procedure view.

WM has limited capacity. Even if experts of both disciplines switch between object view and procedural view, this does not mean that they can use the two views simultaneously, or that they are able to switch while communicating with colleagues.

We found different references to mental models in the literature concerning cognitive aspects of (OO) programming and user-database interaction. These are: (1) the notional machine, simulating an idealized computer, (2) the object view, where the individuation of (problem domain) objects guides the design activity, emphasizing static aspects of the solution, (3) the procedure-centred view, emphasizing the dynamics of a program and (4) a set-theoretical model, describing the problem domain in abstract terms and thus defining the database world.

We hypothesize differences between groups in the instantiated mental models used to handle a concept that is fundamental to both disciplines: the "object".

5. A first experiment: how do professionals understand their systems?

We designed a teach-back protocol to elicit information about the participants' mental models and recruited students of comparable age and level of education, enrolled in different computing curricula. The participants' answers are scored along categories, derived from the four types of mental models mentioned in section 4. Three raters are involved in this experiment. Two are currently lecturing Computer Science in Dutch higher education, one is senior designer of documentation for scientific software. All the raters have a background in computer science and hold a Master's degree.

The protocol was tested in a pilot study with 6 participants, all lecturers of Computer Science in higher education. Based on the feedback, textual changes were made to the questions.

5.1 Questions

The protocol is introduced by a brief case description, an online Bookstore (see Figure 1).

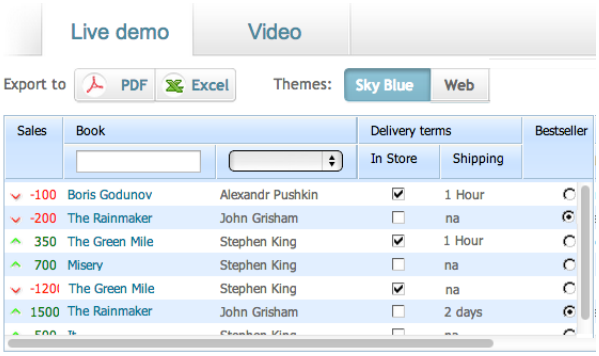

Figure 1: The context of the teach-back questions: an online Bookstore

The case description states that only information about books is relevant for our purposes, and that, therefore, the rest of the data is skipped. This results in a dataset with repeating rows having the same state. Participants are asked to count and describe the "Book-objects" they distinguish. No indications are provided for the choice of a theoretical context.

This situation is, in fact, ambiguous: the question can be answered by describing: (1) books in the bookstore's assortment (objects in the problem domain), (2) Book-objects in permanent storage (in the programming domain) or (3) instances of Book-objects (in the programming domain)

Participants are instructed to explain in writing to an imaginary fellow student: (1) How many different Book-object they see and what these objects are, (2) What happens if the system is asked to produce additional information about one of the books. Up to now, we have been concentrating on the analysis of the first part of question (1): "How may different Book-objects do you see?"

5.2 Scoring categories

To establish the number of different Book-objects, the rater scores one of the following answers:

- 4 objects (or 5, if the participant counts the last row, which is only partially visible),
- 6 objects (or 7, if the participant counts the last row),
- O (a number that is not traceable to Book-objects, or no number mentioned),
- B (both answers "4 objects" and "6 objects" are mentioned and explained).

5.3 Participants

The teach-back questions were answered by four groups of male Bachelor students, enrolled in professional curricula:

- 35 students attending 1st year classes Business IT & Management. Most of them enrolled in a professional curriculum in a computing discipline in 2013, 2 in 2010.
- 19 attending 3rd year classes Business IT & Management. Most enrolled in 2011, 1 in 2010, 1 in 2008.
- 18 attending 3rd year classes Computer Science, field of study Information Engineering. Most enrolled in 2011, 6 in 2010, 1 in 2009, 1 in 2002.
- 29 attending 3rd year classes Computer Science, field of study Software Engineering. Most enrolled in 2011. 4 in 2010.

All groups had attended at least one course "Relational Databases" and one course "Introduction to Programming in Java". The Business IT & Management curriculum emphasizes modelling. The

Computer Science curriculum concerns software development and has two fields of study. Software Engineering puts the emphasis is on programming, Information Engineering on the construction of business solutions. (Studiegids Hogeschool Utrecht, 2012).

5.4 Hypotheses

With this experiment, the following hypotheses are tested:

H1: "there is no significant difference between the conceptualization of the notion of "object" reported by 1st and 3rd year students Business IT & Management".

H2: "there is no significant difference between the conceptualization of the notion of "object" reported by members of the following groups: 3rd year students Business IT & Management, 3rd year students Information Engineering and 3rd year students Software Engineering".

6. Results

The following categories were scored:

- 4: 47 participants. Many of them explain their choice ("There are 6 Book-objects, but two of them occur twice"), indicating that they are counting sets of objects.
- 6: 26 participants. Explanations vary from "n rows, n Book-objects" to "6 Book-objects. Some occur twice, but they are different objects" and "I am thinking OO-Java".
- O: 28 participants.

Category "B" was not scored in our samples. The answers are summarized in Table 1 and Table 2.

	4	6	O	n
1st year Business IT & Management	19	6	10	35
3rd year Business IT & Management	7	1	11	19

Table 1: number of Book-objects counted by students Business IT & Management

We found differences close to significance between the samples in Table 1 (chi sqr = 4.87; $p<0.1$) and reject H1. Most students of the 1st year Business IT & Management count 4 objects. They seem to interpret "objects" as problem domain entities or as database entries. 3rd year students seem to be less certain in their interpretation: their answer is scored more frequently "O".

	4	6	O	n
3rd year Business IT & Management	7	1	11	19
3rd year Information Engineering	5	10	3	18
3rd year Software Engineering	16	9	4	29

Table 2: number of Book-objects counted by 3rd year students, sampled by curriculum

We found significant differences between the samples in Table 2 (chi sqr = 19.19; $p<0.01$) and reject H2. We conclude that future Information Engineers seem to interpret "objects" as instances: the answer "6 objects" is predominant. Future Software Engineers show the opposite preference and count 4 objects. Business IT & Managements students seem to elude the question.

7. Preliminary conclusions.

The analysis of the experiment has just started. It is too early to draw full conclusions about the conceptualization of the notion of "object". We can nevertheless underpin some observations.

Our investigation shows at least two ways to characterize the abstract notion of "object" that are currently used by professionals. These characterizations are not compatible and lead to different judgements about "objects". In the same situation, some professionals will identify 4 objects but others will perceive 6. The preference for 4 or 6 objects is not distributed ad random between professionals. We found indications for differences between groups of 3rd year students, enrolled in different computing curricula. Different preferences in the conceptualization of "object" can be a source of communication problems between groups of computing professionals. The lack of agreement about the definition of one of the basic notions of the discipline is alarming, just as the apparent difficulties to recognize this issue and to discuss it.

We suggest that professional organizations and institutes for higher education establish a refined terminology on this subject, with different terms for instances, stored objects and problem domain objects. This will not solve the Object Relational impedance mismatch, but will help professionals to discuss it without risking to add severe misunderstandings to the problems they are trying to solve.

8. Acknowledgements

We thank the Hogeschool Utrecht and Johan Versendaal for their support; we thank the students and their lecturers for their cooperation.

9. References

du Boulay, O'Shea, Monk, 1989: *The black box inside the glass box; presenting computing concepts to novices,* International Journal of man-machine studies 14: 237-249

Cañas, J. J., & Antolí, A. (1998). *The role of working memory in measuring mental models*. In Proceedings of the Ninth European Conference on Cognitive Ergonomics–Cognition and Cooperation. European Association of Cognitive Ergonomics (EACE): Rocquencourt, France.

Chan, H. C., Wei, K. K., & Siau, K. L. (1993). User-database interface: the effect of abstraction levels on query performance*. *MIS Quarterly*, *17*(4), 441-464.

Détienne, F. (1997). Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams. *Interacting with Computers*, *9*(1), 47-72.

De Haan, L., & Koppelaars, T. (2007). *Applied mathematics for database professionals*. Apress.

Ireland, C. (2011, January). Exploring the Essence of an Object-Relational Impedance Mismatch-A novel technique based on Equivalence in the context of a Framework. In *DBKDA 2011, The Third Int. Conference on Advances in Databases, Knowledge, and Data Applications* (pp. 65-70).

Johnson-Laird, Ph., *Mental Models*, chap. 12 from: The foundations of Cognitive Science, M.I. Posner (Ed.), Cambridge, MA MIT Press, 1989

Norman, D. (1983). *Some observations on mental models*, chap. 1 from: Mental models, Gentner, D., & Stevens, A. L. *(Eds.). (1983). Psychology Press.*

Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, *19*(3), 295-341.

Reisner, P. (1981). Human factors studies of database query languages: A survey and assessment. *ACM Computing Surveys (CSUR)*, *13*(1), 13-31.

Robins, A., J. Rountree, N. Rountree (2003): *Learning and teaching programming: a review and discussion*, in: Computer Science Education 2003, vol. 13, pp. 137-172

Studiegids Bacheloropleidingen Institute for ICT 2012-2013 (2012), Hogeschool Utrecht, Utrecht, september 2012

van der Veer, G. C. (1990). *Human-computer interaction: learning, individual differences, and design recommendations*. PhD thesis, Vrije Universiteit, Amsterdam.