Intuition is a powerful tool that helps us navigate through life, but it can get in the way of more formal processes.

HOW INTUITIVE IS OBJECT-ORIENTED DESIGN?

By Irit Hadar and Uri Leron

The object-oriented programming paradigm was created partly to deal with the ever-increasing complexity of software systems. The idea was to exploit the human mind's natural capabilities for thinking about the world in terms of objects and classes, thus recruiting our intuitive powers for building formal software systems. Indeed, it has commonly been assumed that the intuitive and formal systems of objects and classes are similar and that fluency in the former helps one deal efficiently with the latter. However, recent studies show that object-oriented programming is quite difficult to learn and practice [1, 3, 7]. In this article, we document several such difficulties in the context of experts participating in workshops on object-oriented design (OOD). We use recent research from cognitive psychology to trace the sources of these difficulties to a clash between the intuitive and analytical modes of thinking.

Recent research in cognitive psychology shows that people consistently make mistakes on simple everyday tasks, even when the subjects are knowledgeable, intelligent people, who undoubtedly possess the necessary knowledge and skills to perform correctly on those tasks. The source of these mistakes is often shown to be the insuppressible influence of intuitive thinking. This research, the *heuristics and biases program*, has been carried out by Kahneman and Tversky and others during the last 30 years, and has led to Kahneman's receiving the 2002 Nobel Prize in economics.[1] In his Nobel Prize lecture, Kahneman opened with the following story:

A baseball bat and ball cost together one dollar and 10 cents. The bat costs one dollar more than the ball. How much does the ball cost?

Almost everyone reports an initial tendency to answer "10 cents" because the sum $1.10 separates naturally into $1 and 10 cents, and 10 cents is about the right magnitude. Indeed, many intelligent people yield to this immediate impulse: 50% (47/93) of Princeton students and 56% (164/293) of students at the University of Michigan gave the wrong answer [2, 4].

What are our mind's mechanisms that may account for these empirical findings? One current influential model in cognitive psychology is *Dual-Process Theory* [4, 10, 11]. According to this theory, our cognition and behavior operate in parallel in two quite different modes, called System 1 (S1) and System 2 (S2), roughly corresponding to our common sense notions of intuitive and analytical thinking.

These modes operate in different ways, are activated by different parts of the brain, and have different evolutionary origins (S2 being evolutionarily more recent and, in fact, largely reflecting *cultural* evolution). S1 processes are characterized as being fast, automatic, effortless, unconscious, and inflexible (difficult to change or overcome). In contrast, S2 processes are slow, conscious, effortful and relatively flexible. In addition, S2 serves as monitor and critic of the fast automatic responses of S1, with the "authority" to override them when necessary. In many situations, S1 and S2 work in concert, but there are situations (such as the ones concocted in the heuristics and biases research) in which S1 produces quick automatic non-normative responses, while S2 may or may not intervene in its role as monitor and critic.

A brief analysis of the bat-and-ball data can demonstrate the usefulness of dual-process theory for the interpretation of empirical data. According to this theory, we may think of this phenomenon as a "cognitive illusion" analogous to the famous optical illusions from cognitive psychology. The surface features of the problem cause S1 to jump immediately with the answer of 10 cents, since the numbers one dollar and 10 cents are salient, and since the orders of magnitude are roughly appropriate. The roughly 50% of students who answer 10 cents simply accept S1's response uncritically. For the rest, S1 also jumps immediately with this answer, but in the next stage, S2 interferes critically and makes the necessary adjustments to give the correct answer (five cents).

Recently, a similar phenomenon has been found in advanced mathematical thinking, with college students learning abstract algebra [6]. While it seems natural that people in everyday situations prefer (however unconsciously) quick approximate responses that come easily to mind over careful systematic rule-bound reasoning, students solving mathematical problems during a university course would be expected to consciously train their methodological thinking to check, and override if necessary, their immediate intuitive responses. From these findings we may understand the strong influence intuition, especially its tendency to be influenced by surface clues, has on our thinking. In this article we demonstrate that a similar phenomenon—and a similar explanation—may also hold for OOD tasks carried out by experts in industry.

A note on terminology: We follow Kahneman and

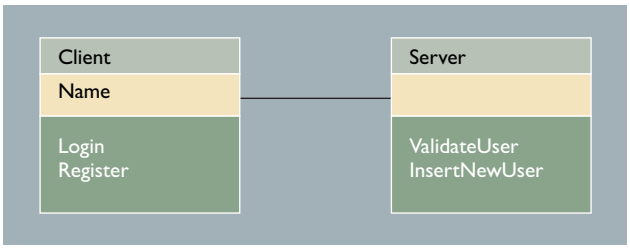
While people in everyday situations prefer responses over careful systemic reasoning, students solving mathematical problems would be expected to consciously train their methodical thinking to check, and override if necessary, their immediate intuitive responses. From these findings we may understand the strong influence intuition.

[1] Tversky unfortunately died several years earlier.

other cognitive psychologists in using "intuition" in its folk meaning of everyday thinking. This meaning is elaborated in the description of System 1, and is mainly used in contradistinction to analytical thinking or to reasoning. The title of this article should thus be understood as an inquiry into the nature of the gap between the everyday "natural" meaning of objects and categories vs. their formal meaning in OOD. It should further be noted that intuition may have different meanings in different contexts. For example, our use of the term is quite different from the way a mathematician might use it when he or she says: "I had the intuitive idea of the proof long before I was able to complete the formal proof."

INTUITIVE THINKING IN OOD

OOD is a complex domain, requiring formal training and effortful thinking, which is just the kind of process System 2 would be expected to appropriate. However, our research indicates that here too the automatic, quick, and effortless operation of System 1 may hijack software developers' attention and lead them to decisions that are not adequate and may even clash with their own knowledge.



A design of authorization system.

We discuss several examples of this phenomenon exhibited by experienced software developers in industry while practicing design activities, and explain them in light of the dual-process theory. We invoke this theory in the domain of OOD in an attempt to understand the relatively elementary mistakes we observed in the responses of intelligent capable professionals, even in cases when they have the necessary knowledge to avoid such mistakes.

Our observations took place within advanced UML workshops [8] conducted in the industry. During these workshops the participants were asked on several occasions to analyze simple design tasks. The participants worked on these tasks either individually or in small groups and their solutions were subsequently discussed within the whole group. Our data includes the written solutions of the participants in the workshops, documentations of their group discussions as observed and documented by the researchers, and transcripts of class discussions. The research population included 41 software developers with experience of 2–12 years in OO development. Because our objective was to describe a complex situation in its natural settings and its full complexity, we have used the qualitative research paradigm [12], which focuses on case studies for obtaining specific insights rather than on large populations, simplified experiments, and statistical methods for discovering universal laws. (This is analogous to the methods used by anthropologists studying unfamiliar cultures.) During the research we documented, videotaped, and analyzed many relevant incidents and processes. The data analysis included coding the data obtained, and characterizing and classifying it to emerging categories. The full research findings and evidence will be described elsewhere. Here, we provide a selection of examples to demonstrate our findings.

Confusing the direction of inheritance. A design task concerning a hotel reservation system was presented for a discussion to a group of experienced engineers participating in a UML workshop. The instructor suggested using three classes (email, fax, and phone), to represent the three corresponding modes of entering reservations. The possibility then arose of using inheritance relations between concrete classes to exploit shared functionality and features, such as checking the availability of a room.

For example, the class fax could inherit from the class email, since a fax object requires more handling (such as scanning and digitizing), hence has more functionality, than an email object.

Instructor: Under the restriction that for now we only use these three classes, can any of these classes inherit from another class? Can we use the fact that they have many things in common?
[The participants hesitate]…
Instructor: For example, fax is like email, only with a few more tests.
Dan: Email inherits from fax, because email is the same as fax, only with fewer tests.
Instructor: So, email has less functionality than…
Dan [hastily]: Oh, right, it should be the other way around.

In view of this and similar observations, we presented a group of 10 software developers with a similar question in order to check this phenomenon more directly. The answers were divided 5:5 between the two possible directions of inheritance. Significantly, as in the bat-and-ball and in Dan's case, the participants who chose the wrong direction required only a small nudge (with no informational or explanatory content) to quickly change their mind.

Analysis: All the participants in the research have

several years of experience in OO software development. Why do intelligent and experienced professionals have difficulties with such an elementary issue? We propose that the same mechanism used by Kahneman to explain the bat-and-ball phenomenon is also in operation here. Specifically, S1 with its quick and effortless operation "hijacks" the thinking process and produces a response that seems roughly appropriate, while the slow and effortful S2 remains dormant. This analysis gets additional support from the observation that the small cue offered by the instructor didn't teach the participant anything new, only served to wake up S2; the necessary knowledge was there all along, but the dual system analysis is needed to explain why it was not mobilized.

Under the demands of abstraction, formalization, and executability, the formal OO paradigm has come to sometimes clash with the very institutions that produced it.

Why would S1 and S2 clash about the meaning of inheritance? In people's everyday intuition (S1), inheritance is about transferring "stuff" (such as property or money), and the direction is usually from the person who has more to the one who has less. For example, in an informal poll we asked students, in the context of OOD, what is the relation between a doctor and a paramedic in an ambulance? A typical reaction was, "paramedic inherits from the doctor because the doctor has more qualifications." Similarly, we predict that most people would say that a student "inherits" from the professor (because the professor has more knowledge) and not vice versa. But in the OOD formalism (S2), the reverse is true: the class with more functionally inherits from the one with less.

Difficulties in identifying objects. One of the first tasks in OOD is "carving a given scenario at its joints" in terms of objects and classes. In one of the workshops the participants were asked to design an authorization system that will route users as follows:

- An existing user will login into the system.
- A new user will register and receive authorization.

A typical design would look like the accompanying figure. The following discussion took place while the participants were working in pairs on the task.

Ron: Let's define login and register as objects.
Sharon: Do login and register seem like objects to you?
Ron: Why not?
Sharon: An object is a client, for example.
Ron: Client is also an object. Login and register are activated and operate within the system; therefore they can be defined as objects.
Sharon: I've never seen an object login.
Ron: Don't worry, it will be okay. You'll see how I design the system; it will be just fine.
Sharon [hesitates, at last reluctantly giving in]: Okay, fine, although it doesn't sound good.

Analysis: Ron's decision is a typical S1 behavior, similar to that observed in the bat-and-ball task. In searching for objects he is influenced by the surface features of the task (the salience of the terms login and register in the task description) rather than its essential (though implicit) components. Unlike the bat-and-ball phenomenon, Ron requires more than a nudge to change his mind, which seems to imply that his S2 knowledge in this regard is not too firm either.

Sharon, in contrast, seems to have a firmer sense of the right objects, but this too is S1 knowledge, in the sense that she cannot explain her choice. Her attempts at convincing Ron involve expressions like "I've never seen an object login," and "it doesn't sound good," which show that she relies on her vast past experience (S1) rather than on analytical rule-based reasoning (S2). Sharon's example, in contrast to the other examples presented in this article, demonstrates how using intuition may in fact contribute positively, even in situations of formal problem solving.

CONCRETIZING ABSTRACT CLASS

Confusing characteristics of abstract and concrete classes. Abstract class is a class with at least one virtual function. Thus one can't instantiate concrete objects directly from an abstract class, but only through a (concrete) inheriting class. In this example, Rebecca chose to define an abstract class car and the following discussion ensued.

Rebecca: Let's say car is an abstract class. Then, in one design I can inherit from it Chevrolet and Rolls-Royce, and in another design I will instantiate an object car with manufacturer value Chevrolet.
Instructor: Is car an abstract class?
Rebecca: No, yes, that's not the point...

In a subsequent interview with Rebecca, the researcher probed the matter further.

Researcher: Rebecca, what did you mean by the car example?
Rebecca: I just tried to show that there are two design possibilities using an abstract class, but I got mixed up.
Researcher: What was the problem?
Rebecca: I wanted to show that you can instantiate objects with parameters instead of using inheritance tree… but it didn't work out.
Researcher: Why?
Rebecca: Because the moment I instantiate objects, I cannot define the class as abstract.

We note that this was not an isolated case. While it seems that the participants in this study recognize the distinction between abstract and concrete classes in theory, several cases were observed where they referred to abstract classes as if they had the characteristics of concrete classes. Even in some written solutions, we found cases where an abstract class was defined but was subsequently used as a concrete class.

Analysis: Rebecca knows the difference between concrete and abstract classes, but this is S2 knowledge. Our interpretation of how S1 worked in this example follows from the dual nature of the relationship between the natural and the formal conceptual framework concerning categories and objects. On the one hand, OOD builds on the intuitions of the natural concepts, but on the other hand, the natural system sometimes clashes with the formal one. We propose that this is what happened in Rebecca's case. Specifically, in the natural categorization system [5], there is no parallel for the formal OOD concept of abstract class (a class from which no concrete objects can be instantiated). Hence, when Rebecca's S2 was not on guard, her S1 took over and slipped from abstract to concrete class. As before, a small nudge was enough to wake up S2 and lead Rebecca to make the necessary distinction.

Identifying software development with coding. Coding is an important software development activity, but other no less important activities contribute to software development, such as requirements analysis, design, and testing. We observed participants underweighting these other activities, to the extent of identifying software development with coding.[2] The following discussion occurred in an interview regarding time invested in different activities:

Ann: Most of the time I was occupied with development.
Researcher: What do you mean development?
Ann: You know, writing the code. For me coding and developing are the same thing, even though I know this is not correct.

Analysis: Ann's first automatic response, that developing is the same as coding, is an S1 response. S1 consists of what is most accessible and what comes most easily to mind; here the view of development as coding comes to mind, presumably because the code is the final and most tangible product of the whole process, while the other components (such as design and requirement analysis) are less conspicuous. The interviewer's question served as a nudge that woke up S2, hence the utterance: "though I know it is not correct." In fact, her second pronouncement is a good demonstration of an actual clash between the two systems: S1 expressing the view that "coding and developing are the same thing," but simultaneously, S2 objecting that "I know it is not correct."

HOW INTUITIVE IS OO DESIGN?

So, how intuitive is OOD? Well, in a certain sense it indeed is intuitive: our cognitive system certainly makes extensive use of objects and categories, on which this paradigm is built. However, as often happens in the evolution of formal systems, this relationship has a flip side [9]. Under the demands of abstraction, formalization, and executability, the formal OO paradigm has come to sometimes clash with the very intuitions that produced it. Thus, while objects, classes, and inheritance certainly have an intuitive flavor, their formal version in OOD is different in important ways from their intuitive origins.

Dual-process theory, imported from contemporary cognitive psychology, highlights the underlying mechanism of those situations where our intuitions clash with our more disciplined knowledge and reasoning. Or, put in Kahneman's words [4]: "Highly accessible features will influence decisions, while features of low accessibility will be largely ignored. Unfortunately, there is no reason to believe that the most accessible features are also the most relevant to a good decision."

[2]This observation was obtained in a joint study with Peleg Yiftachel.

Indeed, we have seen that, under the force of these general cognitive mechanisms, deciding on appropriate objects, classes, and relations is sometimes influenced by irrelevant surface clues or everyday meanings of these concepts, thus leading to inappropriate choices. Intuition is a powerful tool, which helps us navigate successfully through most everyday tasks, but may at times get in the way of more formal processes. We hope this article may contribute to better understanding of this problem, and point the way to thinking about its resolution. C

References

1. Armstrong, D.J. The quarks of object-oriented development. *Commun. ACM 49*, 2 (Feb. 2006), 123–128.
2. Gilovich, T., Griffin, D., and Kahneman, D., Eds. *Heuristics and Biases: The Psychology of Intuitive Judgment*. Cambridge University Press, 2002.
3. Holmboe, C. A cognitive framework for knowledge in informatics: The case of object-orientation. *ITiCSE'99 Conference Proceedings*, (June 1999), 17–20.
4. Kahneman, D. (Nobel Prize Lecture). Maps of bounded rationality: A perspective on intuitive judgment and choice. In Les Prix Nobel, T. Frangsmyr, Ed. (2002), 416-499; www.nobel.se/economics/laureates/2002/kahnemann-lecture.pdf.
5. Lakoff, G. *Women, Fire, and Dangerous Things. What Categories Reveal about the Mind*. The University of Chicago, 1987.
6. Leron, U. and Hazzan, O. The rationality debate: Application of cognitive psychology to mathematics education. *Educational Studies in Mathematics 62*, 2 (2006), 105–126.
7. Morris, M.G., Speier, C., and Hoffer, J.A. An examination of procedural and object-oriented systems analysis methods: Does prior experience help or hinder performance? *Decision Sciences 30*, 1 (Winter 1999), 107–136.
8. OMG Object Management Group. *UML Notation Guide. Version 1.3*, 1999.
9. Paz, T. and Leron, U. The slippery road from actions on objects to functions and variables. *Journal of Research in Mathematics Education*; http://edu.technion.ac.il/Faculty/uril/papers/Paz_Leron_Actions_vs_Functions.pdf.
10. Stanovich, K.E. and West, R.F. Individual differences in reasoning: Implications for the rationality Debate. *Behavioural and Brain Sciences 23* (2000), 645–726.
11. Stanovich, K.E. and West, R.F. Evolutionary versus instrumental goals: How evolutionary psychology misconceives human rationality. Psychology Press, 2003, 171–230.
12. Strauss, A. and Corbin, J. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage, Newbury Park, 1990.

Irit Hadar (hadari@mis.haifa.ac.il) is a lecturer at the Department of MIS, University of Haifa, Israel.
Uri Leron (uril@technion.ac.il) is a Churchill Family Professor (Emeritus) of Science and Technology Education at the Technion—Israel Institute of Technology, Haifa, Israel.

DOI: 10.1145/1342327.1342336