

Levels of abstraction: student understanding of object-oriented programming concepts in context

Anna Eckerdal
Uppsala University

Robert McCartney
University of Connecticut

February 6, 2005

Research questions:

1. How abstractly do students think about standard object-oriented programming concepts?
2. Do these levels change when students are working in different contexts?
3. Do these levels change as students become more advanced?

Motivation

The central problem in CSEd/CS1 is: why do students have such a difficult time learning O/O and Java programming? *Any* results that provide insights here are interesting, because this is such an important problem.

Hypothesis: students struggle because they understand key concepts at the “wrong” abstraction levels (i.e., do not match the levels needed to use the concepts effectively, or the levels expected by the instructor.) Personal observation: intro students (2nd semester) deal poorly with abstract concepts—even the best students have a low-level/mechanical view of O/O, so cannot design well at an abstract level. Some work[4] suggests that students try and cope with new concepts by decreasing the abstraction and dealing with it at a more concrete level, [1] looked at abstraction levels used by data structures students (and noted the preference of students for the less abstract). Constructivist models of CSEd[2, 3] seem to suggest that students gradually learn more abstract versions of concepts incrementally.

However, the stated hypothesis would be really difficult to address, so we ask simpler questions:

- what abstraction level of O/O do students work at?
- is this dependent on context in which they are working?
- does this change as they take more courses?

Specifically, here are the proposed aspects of this study:

Approach: design and administer a questionnaire. For some number of concepts, present students with a number of alternative (correct) definitions. Ask students to rank-order them according to how well definitions describe concept. Compare results across institutions and experience levels.

The concepts: Object, class, method, instance, array, variable...

The source of the definitions: textbooks or other references describing the O/O paradigm.

To be, or not to be (in context): a question kicking back and forth between the two proposers is whether to explicitly give students contexts in which to answer questions. On the one hand, it leads to richer (i.e. more) data, and it would be reasonable to view concepts differently at different times, design v. debugging, e.g. On the other hand, it means more questions (longer to answer), and setting multiple contexts explicitly may bias the results toward changing rankings with context. How to deal with this is still under discussion. We give an example of some highly-contextualized questions in the appendix.

What evidence would convince me? There are well-understood tests for rank data that allow comparison of rankings from different groups, and detect if there are significant differences. *If* we observe significant differences, they would be highly credible. If we do not see differences, however, what would that mean? The fact that the subjects are ranking on preference, not abstraction, may lead to more noise in the data—it may be hard to compare two definitions, if one is more and one is less abstract than the preferred one—may bias against finding differences. Having alternatives that differ primarily based on abstraction is necessary; that this is not trivial is illustrated by the answers in the appendix.

Expected implementation

Given the uncertainty over questions and techniques, we plan to start with a pilot study, using a limited number of subjects and institutions. We will use that information to evaluate the approach. If the results are favorable, we will use them to tighten up questions and techniques, then collect data broadly. Data collection and analysis should be fairly easy once we have good questions.

References

- [1] Dan Aharoni. Cogito, ergo sum! cognitive processes of students dealing with data structures. In *Proceedings of the 31st SigCSE...*, pages 26–30, Austin, TX, 2000.

- [2] Mordechai Ben-Ari. Constructivism in computer science education. In *Proceedings of the 29th SigCSE...*, pages 257–261, Atlanta, GA, 1998.
- [3] Ann E. Fleury. Programming in java: Student constructed rules. In *Proceedings of the 31st SigCSE...*, pages 197–201, Austin, TX, 2000.
- [4] Orit Hazzan. How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education*, 13(2):95–122, 2003.

Appendix: Example of heavily contextualized questions

Given a specific design/programming problem (which is given to subjects), answer the following:

For each question, rank the alternative answers from best to worst (*Possible alternatives*: “Choose the three best alternatives and rank them”, or “Choose the best answer”).

1. You have just been introduced to this problem. When you start to design a solution for this problem, you think of objects in terms of:
 - (a) data in a computer memory
 - (b) the different parts my program consists of
 - (c) actions the objects perform during execution
 - (d) how the objects are defined in the classes
 - (e) what the objects represent in the real world
 - (f) the Java language syntax rules
 - (g) the nature of being
2. You have designed your problem and are now coding. When you get a error message from the compiler, you think of objects in terms of:
(*Same answers as 1.*)
3. You run your program. When you get a run-time error, you think of objects in terms of:
(*Same answers as 1.*)
4. You talk to friend who doesn't know anything about computers or programming. You explain objects (in object-oriented programming) in terms of:
(*Same answers as 1.*)
5. Your teacher asks you about your work. When explaining to him/her your solution of the problem given, you think of objects in terms of:
(*Same answers as 1.*)