Proposal to NoPhiCER 2008

How could we investigate students' experiences of software design and relations between models and code?

Jonas Boustedt
Division of Scientific Computing
Department of Information Technology
Uppsala University
SE-751 05 Uppsala, Sweden
jbt@it.uu.se

ABSTRACT

We want to know more about how students experience aspects of software design and problem-solving. In particular, the upcoming research aims to explore how students experience modelling and coding of object-oriented software and the relation between models and code.

We plan to start this phenomenographic study in the autumn of 2008 and it will involve computer science students from one or more Swedish universities. The analysis should be accomplished during spring 2009.

The proposed (tentative) research question is formulated as follows:

> *How do students experience object-oriented software design?*

The research question needs to be operationalized, and I suggest the following aspects to be investigated by phenomenographic interviews and analysis:

- In what ways do students experience design models?
- How do students explain the relations between concepts in models and in code?
- How do students express the motivation for using object-oriented design?

Answers to these questions should possibly help to formulate partial answers to underlying questions about aspects of "programming-in-the-large", implications for teaching, and more.

I look forward to discuss these matters with phenomenographers at NoPhiCER. The proposed discussion questions are labelled "DQ" in the following text.

Why do I think this is interesting?

Object-oriented software design involves the question of how programmers should structure their designs to efficiently produce understandable, maintainable, changeable and reusable software. This is very problematic and challenging from an educational point of view, because it is easy for both students and teachers to see the concrete details with short term goals, but it is much harder to for teachers to explain and motivate advanced abstract concepts with long term goals, especially if the students' views on software design are shrouded in mystery.

I believe that if you go "design first", modelling and design require a good understanding of the meanings of the symbols and idioms of the modelling language itself and their possible interpretations (meanings) in code. And if you go "coding first" (some do it well) it takes a good ability to model "intuitively in the head". In both cases the relations between the model and the implementation are important. I find it very exiting and meaningful to learn more about how students experience this relation.

The results from this study should be interesting for researchers and teachers, and potentially they could be used to analyse and structure lectures, assignments, courses and curricula with consideration to the students' experiences.

Issues to discuss at NoPhiCER

We have all met students who did not care much for design and modelling, preferring "headfirst programming" in the beginning of their education, and for some students it takes a long time before they appreciate "design first" – if they ever get it.

> DQ 1: *How should the phenomenographical research be designed to allow useful application of results?*

Following the theme of modelling software, I plan to investigate the students experiences of relations between various phenomena expressed in symbolic (graphical) notations (such as the UML) and in the used programming language (such as Java). Examples of phenomena are model, aggregation, composition, navigability, inheritance, multiplicity, class/object, packages, implementation of interface, polymorphism, et cetera (see Figure 1).

> DQ 2: *Are the students' experiences of these phenomena interesting for other researchers in the CER community and phenomenographers as well?*

One of my ideas for data collection is inspired from my previous work (see below). The idea is to have stu-

dents doing problemsolving, design and implementation in pairs, thus forcing recordable conversations during their work. In an additional experiment the same students should re-engineer models from given codes. The researcher is present all the time and takes notes. The researcher consults the notes and makes adjustments to a semi-structured phenomenographic interview script and carry out the interview with the students separately. The interviews will be investigated through phenomenographic analysis, but could it also be the case that parts of the recorded conversations during modelling can be analysed?

> DQ 3: *Apart from interviews; how can phenomenography be applied to transcribed discussions between people and how should data collection be prepared in practise?*

Forms for discussion

I want to share my ideas about research questions, data collection, analysis and how to make conclusions. In the discussion I hope to get comments and suggestions from other phenomenographic researchers from the field of CER.

Since I want to get a rich, comprehensive view on students' experiences of modelling and its elements, the design of the study must be planned carefully. I hope that this discussion could be interesting for other researchers that have similar research questions or are interested in similar methods.

Experience and interest in Phenomenography

My research interest is aimed at aspects of students "programming-in-the-large" as opposed to "programming-in-the-small". In my licentiate thesis, I describe a phenomenographic study on students' experiences of working with a "large" software system during a role-play experiment where the students acted as a newly employed programmer at a software company. Their mission was to complete a large software project where the senior programmer had taken ill. By reading the documentation, they should realize that they had to write the code for a new plug-in module.

The role-play was followed by a debrief and a phenomenographic interview that lasted for about 45 – 60 minutes. The students were asked about their experiences of the role-play, their conceptions of the software system and their ideas about the Java interface concept and plug-ins, which was frequently utilized in the software. The experiences of these phenomena were investigated using phenomenographic analysis. The resulting outcome spaces were presented and discussed and some conclusions were drawn.

I wanted the students to be in a state of mind where they had the experiences from the role-play near at hand, making it easy to relate the discussed phenomena to concrete situations and to the specific details of the software.

My conclusions of the effects of the chosen data collection method in that study are:

1. it helped to bring forward aspects of "programming-in-the-large" to the interviews,

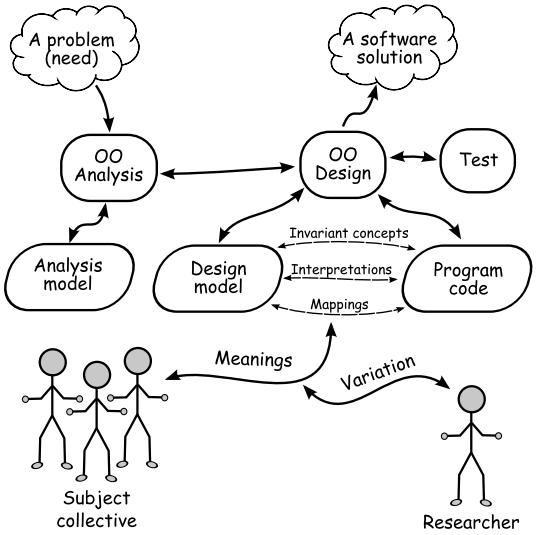

Figure 1: A model of the object-oriented problem solving process and what I intend to study: the relation between the students and phenomena associated with modelling.

2. the students appreciated their participation in the role-play and in the interviews which lead to a friendly and open-minded setting,

3. the interviews were stimulating and the previous role-play helped both the interviewer and the interviewee to stay focused,

4. the conditions for looking into the structural aspects of the studied phenomena was strengthened because the students could relate both generalizations and specializations (examples from the software) in their descriptions.

Some thoughts about learning design

Compared to what students learn in introductory programming courses, software modelling is an activity that requires an ability to move between abstraction levels. The meanings of the symbols that are used can be both ambiguous and deep, however, they can also suggest a very concrete meaning in code. The designer's focus is shifting; from beeing aware of patterns of relations formed by the symbols and what advantages they have, to beeing aware of the concrete effects the symbols and the relations have, implemented in some specific programming language. Before this comes natural, I believe it takes some time and experience

Once perspectives smoothly can be shifted from the concrete to the abstract, beeing concrete about the abstract, and shifted back to the concrete from the abstract – perhaps it is only then the learner can utilize modelling in a meaningful way.