Educators' Strategies for Object Oriented Analysis and Design

Marie Nordström
marie@cs.umu.se
Department of Computing Science
Umeå University, Sweden

Abstract

Almost all research on the teaching of object orientation has been focused on the students and their learning. One important aspects that will affect how object orientation is taught, is the educators personal views on different issues of the paradigm. In this paper we present some results of a qualitative study on educators views on the teaching of object orientation. We specifically focus on how teachers address object oriented design and analysis. Data was collected through interviews with ten educators.

1 Introduction

An important aspect of teaching object orientation to novices is to introduce the students to a problem solving approach. Without knowing how to approach a problem and how to look for suitable objects in the problem domain, it is difficult to acquire skills in object oriented problem solving and programming.

We know very little about the way educators themselves think about object oriented analysis and design. We do not know of the methods used by experienced educators to enhance the introduction to object oriented problem solving and programming. There are, to our knowledge, no studies on these perspectives of teaching object orientation to novices. The lack of previous work in teachers views on object orientation has made this study exploratory in nature.

Discerning different views of educators teaching object orientation to novices, seems critically important due to the implications for student retention, the quality of higher education as well as the quality of the professional training of teachers for upper secondary schools.

The method used to investigate the area is a qualitative approach using qualitative content analysis of semi-structured interviews.

The research question investigated in this paper is *How can educators' views on OOA&D be characterised*?

2 Related Work

It has been argued that object orientation is a "natural" way for problem solving. However, several studies question this claim (Guzdial, 2008), when asked to describe a given (algorithmic) situation, e.g., situations and processes that occur in a Pacman game, non-programmers did not indicate any use of categories of entities, inheritance or polymorphism. It has also been shown that novices have more problems understanding a delegated control style than a centralised one (Du Bois et al., 2006).

Dale (2005) presents the result of a survey asking educators about the most difficult thing to teach in CS1. In the category object-oriented constructs, polymorphism and inheritance were major topics mentioned. Seemingly minor topics reported were: instance methods, instance variables and static variables. Even such basic ideas as user defined classes and objects were considered difficult to teach by some of the respondents. The struggle to find a balance between object orientation and more general programming constructs was also frequently mentioned. Object oriented analysis and design was considered hard to exemplify because of the, by necessity, small examples.

Most of the research involving teachers has been to investigate what they think of their students' difficulties, and on different approaches to teach object orientation (Clancey, 2004; Holland et al., 1997; Kaczmarczyk et al., 2010; Eckerdal et al., 2005). Thompson (2008) has been exploring practitioners perceptions of design characteristics in object oriented programs. The results show a span of five perspectives, from the lowest with a focus on language, to the highest category where the cognitive process is the primary focus. The higher level categories do not ignore technology aspects but see them as taking a subordinate role. Only the two highest levels concerns abstractions, while the lower ones models real world objects.

3 Programming Education

To investigate educators' views on object oriented analysis and design we have chosen to focus on both lecturers at the university level and upper secondary school teachers.

In Sweden, in general, the computing curricula of university programmes with CS majors or minors, contain traditional CS1 and CS2 courses.

The Swedish upper secondary school is entered at the age of 16, and is regulated by the Swedish National Agency for Education (Skolverket (2010a)). Because of this, the syllabi of programming courses in upper secondary school is known, which makes it meaningful to include teachers at this level.

All courses concerning computers and programming are organised in a subject called Computer technology. In Figure 1 the structure and relationships among the programming courses in upper secondary school is shown.

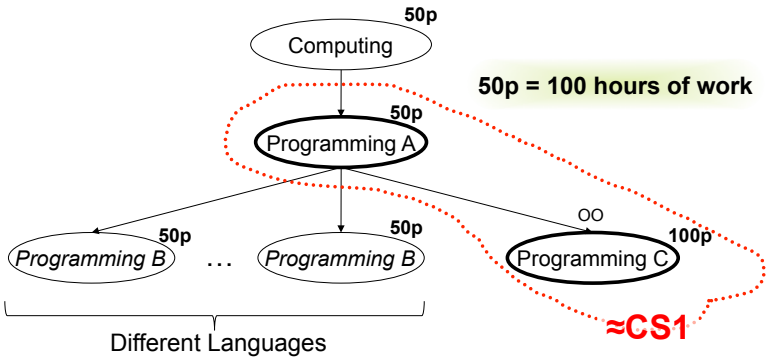

Fig. 1: The structure of programming courses in Swedish Upper Secondary School

Computing provides knowledge of PCs and skills in using software. *Programming A* provides a basic theoretical and practical knowledge of programming. *Programming B* is aiming at theoretical and practical knowledge in a structured programming language and skills in designing algorithms. *Programming C* provides theoretical and practical knowledge in an object-oriented programming language, as well as a knowledge of analysis and design methods. It also provides knowledge of graphical user interfaces, see Skolverket (2010b) for more details.

4 Method

The research question was thematically operationalised according to four themes; the paradigm itself, the concept of an object, examples and the problem solving process of object orientation. Each theme was viewed from three different aspects, the educator's personal view, the educator's view of student difficulties and the educator's choice of methodology to address those difficulties, see Figure 2 and (Nordström, 2010) for further details.

The goal of all qualitative research is to understand a phenomenon. Shank and Villella (2004) use the metaphor of a lantern to describe qualitative research. In this particular study the purpose was to explore and investigate different ways of thinking about object orientation among educators.

The data for this study has been collected through semi-structured interviews and analysed through qualitative content analysis.

4.1 Sample

In all, 10 interviews were conducted, 6 with teachers from upper secondary school (students at the age 16-19) and 4 with lecturers at the university level. The group of interviewees consists of nine men and one woman, all, except one, with many years experience of teaching and experience with object orientation.

	Teacher's personal view on concept	*Teacher's view of students difficulties*	*Teacher's choice of methodology*
	Characteristical	**Problematic**	**Teaching-practice**
Paradigm (OO)	What are the characteristics of OO? What is most important to stress?	What about OO is most difficult to internalise?	How is OO presented, as paradigm?
Concept (Object)	Ideal objects, how are they defined?	What is perceived as difficult about objects?	How does a displayed object typically look?
Examples	What is characterstic of a good example?	What makes an example difficult for students?	How are examples chosen and/or designed? What characteristics are prioritised?
Process (OOA&D)	What is characteristic for the problem-solving approach?	What do students find difficult in OOA&D?	How is OOA&D introduced and practised?

Fig. 2: Interview guide.

The educators participating in this study show diverse backgrounds. The demographic data shown in Figure 3 consists of what type of degree the respondent holds, if object orientation has been acquired through formal training or through own studies. The figure also shows whether the interviewee teacher at upper secondary school (USS) or university. The relative size of the schools as given as well as the number of years of experience in teaching programming, regardless of paradigm.

The demographics of the respondents are shown in Figure 3, and the content described below.

ID	Degree	OO	School	Size	Exp
R1	T	Self	USS	M	18
R2	T*	Formal	USS	S	2
R3	Bach CS+T	Formal	USS	S	11
R4	T	Formal	USS	L	11
R5	T	Formal	USS	L	13
R6	T	Self	USS	M	13
R7	PhD IS	Formal	U	M	11
R8	Master CS	Formal	U	S	11
R9	Bach. IS	Formal	U	S	16
R10	PhD CS	Formal	U	L	5

Fig. 3: Demographics of Interviewees. T*: R2 is on the way to a teaching degree at the time of the interview.

ID All interviewees are identified by an simple code, R1-R10.

Degree Knowing that the recruitment of CS-teachers for upper secondary school

is difficult, it was interesting to collect information on the formal degree of the respondents. Degree-abbreviations: T=trained teacher, CS=Computer Science, and IS=Information Systems. T* is on his/her way to a teachers degree, but not graduated at the time of the interview.

OO Furthermore, we collected information on how the interviewees had gained their competence and skills in object oriented problem solving and programming, whether they had formal academic training or were autodidacts.

School The first six respondents work in upper secondary schools (USS), and the last four lecture at university-level (U).

Size It is always a risk that small institutions have more restrictions on their courses, e.g. having students from very different programs in the same class, which may affect the teachers working conditions. Therefore, we made an effort to have Small (S), Medium (M) and Large (L) size schools/ universities represented in the population, which was successful.

Exp The last column of Figure 3 shows the respondents' experience, in years, in teaching programming (*Exp*).

It was not easy to find women teaching object orientation, so we are grateful to have one woman among the respondents. Sample size for qualitative studies is often discussed, and Sandelowski (1995) concludes that the quality of information obtained per unit is the most critical measure. Sample size is difficult to determine and one recommendation is to proceed until analytical saturation is received. Another recommendation for this particular kind of study is to include about six to ten participants (Sandelowski, 1995; Morse, 1991).

All of the upper secondary school teachers, with the exception of R3, are trained teachers in maths and/or physics. Another background not uncommon in Sweden, is to have a bachelors degree in some major subject and then to add courses for the fulfillment of a teachers degree (e.g. R3). This variety in teacher background in upper secondary schools is due to the fact that Computer Science is not recognised as a subject within the teacher training programmes in Sweden. The lack of trained CS-teachers makes it common for schools to assign science teachers, even without formal CS training, to teach programming courses. They are often autodidacts and on many schools the sole teacher in this subject. CS being a young discipline it is not unusual for university educators to teach before and during their PhD-studies. One of the university lecturers in this study earned a PhD in Chemistry before switching to CS.

4.2 Interviews

The interviews lasted in the range of 45 minutes to 1 hour and 16 minutes. The interviews were all conducted by the author, in Swedish. A verbatim transcription was conducted by, or supervised by, the author, using Transcriva. All interview quotes throughout the present paper have been translated by the author.

In an attempt to limit the effect the interviewer might have on the interviewee, the vocabulary was kept at a non-formal level, to avoid any unnecessary influence or intimidation of the interviewee. The interviewer is always part of the research, and the relationship between the interviewer and interviewee affects the outcome of the interview.

In all cases, the interview was performed at a locality of the interviewees choice, on most occasions in their office. Every interview starts with the interviewer asking the interviewee to describe his/her background, how he/she came to this point in his/her professional life, teaching object orientation to novices.

4.3 Analysis

The analysis has been done using qualitative content analysis (Hsieh and Shannon, 2005; Forman and Damschroder, 2007). Content analysis is a widely used qualitative research technique, particularly in health studies (Graneheim and Lundman, 2004; Hsieh and Shannon, 2005; Elo and Kyngas, 2008). Current applications of content analysis show three distinct approaches: conventional, directed, or summative. They are all used to interpret meaning from the content of text data. The major differences among the approaches are coding schemes, origins of codes, and threats to trustworthiness. In *conventional content analysis*, coding categories are derived directly from the text data. With a *directed approach*, analysis starts with a theory or relevant research findings as guidance for initial codes. *A summative content analysis* involves counting and comparisons, usually of keywords or content, followed by the interpretation of the underlying context (Hsieh and Shannon, 2005).

In this study the conventional approach has been used, because of the lack of previous studies. The primary objective is the manifest view of object orientation investigated through a number of different aspects. A thorough description of the study as well as results concerning educators personal view on object orientation, objects and examples for object orientation can be found in (Nordström, 2010).

To be able to return to the original record for any statement, at any time during the analysis, they were all given an identification tag [*id_ row*], where *id* is the respondents identification (see Figure 3), and *row* is the row number of that particular transcript.

5 Results

In the present paper, two aspects are analysed: *Educators choice of methodology for introducing OO* and *Educators choice of methodology for teaching OOA&D*.

5.1 Introducing the Paradigm

To investigate how the paradigm was introduced, information had to be collected throughout the interviews. A more direct question would often be perceived as asking about objects or programming in general.

Three categories of strategies for the introduction of object orientation as a problem solving approach have emerged from the interviews. They can be characterised as: *Building a world of objects*, *Induced by contexts, databases and concepts*, or *Not addressed*.

Building a world of objects

In upper secondary school the students have already been introduced to programming through the procedural/imperative paradigm. Some of the teachers utilize this prerequisite knowledge to show the difference in approach using the object oriented paradigm.

R4_182: *eh, i use, i use an example close to them, that i, that we do it differently, [...] so far we have been working with functional programming, eh and..., and try to show what the difference is. eh.. and that in object orientation you explain this with the aid of objects, a world. that, for example, that we are to describe a forest in procedural programming then you see things that might happen in the forest, functions, birds sing, trees grow, leafs are falling and so on. in object oriented programming you see the objects that make up the forest, trees, stones, birds, and so on... that i try, something like that, that was the forest before and this is the forest now.*

In many university programmes the introduction to programming is done directly in the object oriented paradigm. In general, R10 and R7 apply the most theoretical approach.

R10_164: *Well... I was going to say that the first thing I do, usually is.., eh... an example that I use.. an enclosed field where it, eh.. where there are different types of beings. There are carrots that grow, and bunnies that jump around and a wolf maybe that chases the rabbits and things like that. So then, then we have different types of objects interacting in this field.*

R7_527: *Well... I am not sure actually, I told before that I try to work scenario ... based, or how I should phrase it, I try to put things into a potential...*

There are however not many instances in these interviews, where the educators explicitly attempt to illustrate object orientation. Much more common is the use of contexts that force some kind of object oriented solution.

Induced by contexts, databases and concepts

R8 avoided discussing how he/she introduces object orientation, using different excuses, this and that being exceptions for a particular course he/she is teaching, so finally the interviewer poses a direct question:

I_194: *but since it has been decided to introduce object orientation, you must have an idea of what that means.*

R8_197: *Yes right, but then it is, the main, the basic idea so to speak. That is, that I want to introduce classes and objects, and try to make small examples where object interact to solve a certain problem, Eh.. and... I don't think I succeed because this is a very small part of the course, eh...*

Using contexts like games seem to be a popular way to "show" the students what object orientation is.

R8_209: *In a way, you might say that object orientation presents itself. The students do a project, because, they make a project, many... and get to chose [project] for themselves, more or less. But since I hint about what to do it ends with them building small games. And that is smart to, then they use an existing module, that enhances this game construction. Eh, and then inheritance show up naturally, they have to inherit from some module classes, so that they have existing Sprite-classes that they inherit to their own Sprite and particularly, well and things like that, so it becomes kind of a drill in , in certain object oriented concepts in that project.*

One of the educators is consistently using databases as a point of departure, even when implicitly introducing object orientation.

R5_9: *[This particular programme] demands a different kind of motivation than the Science programme, you can not introduce the concepts theoretically to them, they need to see the practical uses. This is done through databases.*

Not addressed

Most interviewees (eight) do not specifically introduce object orientation as a paradigm.

I_299: *So, you are not making any introduction to object orientation in general, as a problem solving approach? The difference compared to the imperative approach?*

R1_302: *Nae.... I ... no not in that respect.*

Prompted for any particular characteristics that would definitely be considered non-object oriented, R2 answered:

R2_435: *Well, it... But I, yes, no I have not... not decided on any high standards when it comes to that [object orientation], no I haven't.*

R6 is really enthusiastic about objects and object orientation, but still has a hard time finding a way to formulate the essence of object orientation to his/her students:

R6_455: *you're standing there trying to convince them that objects are IT.*

I_458: *And how do you get this idea through to them?*

R6_461: *I think I do that through.. partly through... eh, it is something that has emerged lately, it has... it has not been around for long. [...] you see it almost everywhere [...] it is a reality [...] hopeless to avoid*

One of the educators even admits to having trouble understanding him-/herself:

R9_972: *I don't remember... but... well, yes of course I have... they know it is about Java, so I try to give some background on Java and why it is object orientation and what the goal of object orientations is. Although I had a hard time accepting Java when it it first showed up, at the end of... or when I started to learn some object orientation, I had a hard time understanding the motives why object orientation compared to procedural [programming] that we had then, but I really haven't gotten the answer yet (laughing).*

5.2 Teaching Object Oriented Analysis and Design

Introducing students to object oriented analysis and design can be done explicitly or implicitly, or not at all. The emerging categories for the methodological approach is seen in Figure 4.

Explicit	Implicit
Lexical analysis	Scenariobased
Design Patterns	Metaphors
Design reasoning	Fomal notation (UML)
	Object-rich contexts

Fig. 4: A&D method

Lexical analysis

R8 is the educator most explicitly discussing active objects as an important part of an object oriented solution.

R10_734: *Well I think I am a little towards, maybe that you start from some kind of... eh, well,... how should I phrase it, this thing with nouns, just to have a starting point. And the I am rather fond of analysing the interaction, to work with that.*

Design Patterns

Several of the interviewees mention design patterns. R5 is explicitly working with the Model-View-Controller pattern. To make the students realise the need for a structured approach he/she makes them "box themselves into a corner" before showing them this pattern. Even though not in the explicit way of MVC-patterns, R2 promotes a separation of the core of the programme from the user interface:

R2_270: *Yes we do that... eh, I have designed a couple of assignments, or I have had a couple of assignments that they start out to solve in a pure... in an environment with only text representation and then they have to add GUI's to that. I do think... that it is a part of the problem that you can separate, that the graphical interface is something that is beside or surrounding the problem, and can be solved separately.*

One of the educators uses design patterns as tools for object oriented problem solving:

R10_647: *[...] we talk about some, some chosen design patterns, I have chosen a couple that I find, eh,... that, that in some sense deals with the more general ideas of object orientation. Eh, like for instance specialisation.*

Design reasoning

The most commonly mentioned way of handling analysis and design is talking while doing. The educator is using his/her own work during teaching as illustration. This is not done in a formal way, but conveyed through discussion of different approaches and decisions while developing a solution to a problem.

R8_563: *Well not that much, well. I don't think, well I don't think that class diagram is that interesting when there is only one class. But it, eh, then you can think and write a little about what, what things we need to know about this Elevator, and then, eh, then I connect the computer and try to run, to write an example corresponding to what's on the blackboard.*

Collecting ideas from the students, makes them active and part of the implicit design process:

R4_266: *[...] In fact, I ask the students what we should do, for instance in a school context. Well, well, Student then, and what.. and then it pops up what kind of attribute a Student should have. And what a Student can do and so on, and what happens if you study? Well, IQ might increase and so on. And so on, and well... as naive as possible [...] they have to participate and it should be as simple as possible.*

Gently guiding the students through problem solving, R10 expresses a strategy of "thinking with the students".

R10_512: *Eh, in this course I also work a lot with me just standing there with the exercises, and I don't bring prepared solutions, because I want to walk the students through the process of thought. Or that they should contribute with input, and from that input we do the thinking process.*

R10 also tries to show several solutions and discuss differences, and consequences of different approaches. This is also to encourage students to try out different ideas, to be a little bit more creative without knowing where it will end. R10 wants to encourage them not to be afraid to try their ideas.

R10_980: *[...] Since I do it on the board [solves problems] and collect ideas from the students, and kind of, someone says something, then "well, then you thought in that direction instead, well then this is the result" and then you hear "well you thought about it this way", and then you really confirm and legitimise different ways of doing it. And encourages the students to do not one, but three solutions.*

R10_986: *and then, then when you have made three, then you start thinking if anyone of them is better than the others.*

Less articulate, in terms of strategy, but still with a conscious goal to foster design:

R2_162: *Well, well I try to help them as best as I can, how to think and to isolate... eh, attributes and phenomena.*

Few of the interviewees uses an explicit method for the introduction of object oriented analysis and design. However, looking at what and how they are describing their work, some implicit strategies for analysis and design show up.

Scenario based

Trying to make sure that problems and exercises are non-artificial, R7 seeks problem domains that are realistic from a software developing point of view, and tries to formulate events in this context.

R7_527: *[...] I try to work from scenarios, or what to call it, try to put things into a potential context*

Even more clearly, R5 expresses that the design is defined from possible events in the system.

R5_37: *Well, this is done through cases, you write down all cases there are and plan for what you need and fetch and so on.*

Metaphors

Some of the educators deliberately uses the metaphors of clients and service providers, to convey an outside view of objects. What can be expected from the clients point of view and what could be supplied from the providers point of view.

R9_423: *Methods, I often compare them to a service supplied by a consultant. I try to explain it that way. That you.... you order a service from a consultant. OK. And then I move on to this, what the consultant need, the parameters. What the consultant need to accommodate my request. And that is how I...*

In support of object thinking, some educators explicitly discusses the use of `main`, or imagined clients.

R10_872: *In my examples, actually we, eh... I have made a separate class, some kind of client-class kind of.*

R10_884: *It is much about acquiring the feeling that programmes are not necessarily something that is directly used by the one sitting at the terminal, but that it might as well be used by another programme, that the the user must not be a human. [...] to return something does not mean to display it on the screen, but it is something sent to someone who have asked for this value, kind of.*

R2_276: *[...] I have talked a lot about the communication between different parts, eh and that is... discussing object orientation based on well defined interfaces among objects. And then, it helps those pieces as well.*

If one way to support the perception of interacting objects is to take the clients view, then another, maybe complementing, approach might be to *play* the role of an object:

R6_392: *When you get into what an object is and ehh.... how to think about an object.. [...] when you call a method, but you may also think about it in terms of talking to the objects, that you pose a question, that it becomes more of a conversation among objects [...] I play a game pretending to be the object and eh-.. maybe pretend that a student is an object...*

Promoting the idea of active objects, as opposed to passive data containers, R10 humanises the objects:

R10_173: *Let's say that someone would like to move this chair, then this person will have to communicate with this chair-object saying "now I push you" and then the chair will react to this in some way. And then it might check "is someone sitting on me?" because then I'm supposed to react in one way, and if not my reaction will be something else, and so on. But again, then the chair becomes active all of a sudden.*

Formal notation – UML

When it comes to notational support for object oriented analysis and design, surprisingly few interviewees brings this up, and when asked directly, they vaguely refer to lack of time and focusing on the "core". Some use UML and others some semi-formal notation of their own.

R7_65: *Yes, we do [use a formal notation] and they learn at least class diagrams in UML and some relations; Association, aggregation and inheritance.*

However, students are seldom required to achieve any notational skills.

R10_275: *But, I have set the requirements to be that if they see a UML-diagram they should be able to recognise and understand it briefly.*

R2_369: *I lecture UML.*

Neither R10 nor R2 require their students to use it in their own development or documentation.

R6_215: *Well, [exemplifying with* `String`*-objects] I drew a circle with data in the middle and functions,* `at`*,* `length` *surrounding on the outside. And then I tell them that what is inside is hidden. And you have to communicate with this text over those functions, something like that.*

Object-rich contexts

One way to compensate for the lack of a methodological approach to object oriented analysis and design is to make sure that the problem domain in itself naturally contains many objects. This way the possibilities of making reasonably good choices when modeling the problem are much higher than when dealing with more general problem definitions. Several of the educators have made a choice to use object-rich contexts. Robots, games and web applications can all be viewed as populated with easily recognised objects. Sometimes this is in combination with databases and graphical interfaces.

R4_236: *Well, when you're dealing with games it becomes much easier to put all those classes into, eh, a game. Now we have this game, these are 3D-models or something, and they, they accept that they understand that this is 3D-models, this may be the camera and so on, and it is much easier for me to argue that if we have this game, we need to have a class that pulls these objects into our game. We need something to control the camera, when you move the mouse or keyboard you have to be able to move the camera and view from different angles, and that, they accept that. And somehow I think it's a little bit fun too and then they almost get along with, well let's do a separate class for this. And then, precisely in this situation it becomes easier for them to accept that you design classes of your own.*

R4 also uses the forest to illustrate the object view of a problem domain, as shown in the quote R4_182 in section 5.1.

R8 teaches a slightly differently organised course, with a shorter introduction to object oriented programming followed by a larger project.

R8_374: *Well they have to, the first day they have to come up with an idea for a project, and then they have a couple of days to move on with the design, or mainly functionality and by that time we have not yet introduced use-cases and things like that, it is more to write...*

Most of the students pick games for their projects:

R8_209: *[...] they can chose almost anything for their project. But since I hint them possible things to do, it usually ends up with them building small games. And the smart thing is that they get to use an existing module [library] that enhances the game building. Eh, and then inheritance is automatically included, they have to inherit from module classes [...] so it becomes an exercise in object oriented concepts in this project.*

R8_377: *But, ... in this you might say that it is quite some thinking to do about what could be needed, and then you could say that, when building small games, well then it pretty much shows itself, that they need a hero and some enemies and a field/track and so on. Eh, and then the rest so to speak, the really difficult parts are handled by this framework with events and things like that, so they don't have to think about that much themselves.*

Other, slightly more restricted, object-rich contexts are hotels, banking and student administrative systems where there are a limited number of entities to deal with:

R9_228: *And then I use for example the results to be formally registered by the student administrator, that could be a class too. Then I move on, from the basics...*

R7_416: *Then I might use an example of a... I have used a hotel scenario where you have a class Booking, and this booking has, besides being placed by someone, it is a booking of something and this something might be a room, and a room then becomes a class of its own, ...*

5.3 Not supporting Object Oriented Analysis and Design

Not supporting object oriented analysis and design does not mean avoiding objects. There will always be objects, and that could in one sense be regarded as supporting analysis and design implicitly through examples. However, in some cases the students got to decide on very few classes of their own. Either the design was given by the lecturer, or through libraries, and in some cases through formal notation or in words. The practices not giving any support for object oriented analysis and design emerging from the interviews can be summarized as: *Data driven*, *Objects supplied*, *Physical objects* and *Design supplied*.

Focusing on databases makes the design entirely driven by the data to be stored and does not leave much room for object oriented design decisions. Using graphical contexts such as games, makes an extensive use of library classes necessary, and the design of objects is to some extent already decided through the library classes available.

Some of the educators use physical objects like chairs and cars, to convey the idea of objects. Used without a motivating context this often results in a static behaviour. It is also common to practice programing skills through implementation of given designs. These designs are sometimes given in UML-diagrams or as a detailed descriptions of named attributes and methods to be implemented.

6 Conclusions

In the educational context we have conditions limiting the possible approaches to teach the subject at hand. What and how an educator does in his/her teaching is depending on several factors. Student group, time frames, curricula to be covered, but also a personal belief of what is essential and what is difficult in the subject, will affect the teaching. The purpose of this study has been to listen to educators describing how they introduce object orientation, and how they introduce object oriented analysis and design.

Not many of the educators make an explicit introduction to the object oriented paradigm. At best they use contexts that are rich in objects, for example games, graphical interfaces, or database applications.

Most of the categories for analysis and design emerging from the ten interviews are used by single educators, and often only mentioned or superficially demonstrated to the students. Only two of the interviewees expressed a more systematic approach to introduce object oriented analysis and design. However, the students were not required to practice these approaches themselves.

For educators in upper secondary school the choice of problem domain is important. To these educators, it is vital to keep the interest and attention of the students with something considered fun, and at the same time find a context that would assist in teaching and learning object oriented problem solving. The two university educators aiming for a more conceptual approach, preferred contexts that are software oriented.

7 Threats to Validity

There is always a possibility that the mere fact that a certain research area is discussed influences the statements of the respondents. Some might have a nagging feeling of being evaluated and might be restricted in their description of certain issues. This is unavoidable.

A conscious choice was to try to use a "neutral" language during the interviews, to avoid intimidating the respondents with a language more formal than the one they would choose themselves to discuss object orientation. However, this might also have been counterproductive and influenced them to use the same wording, instead of their own vocabulary. The object oriented vocabulary has not been a major part of the analysis, and great effort has been made to listen to descriptions rather than exact wording. My point of departure is not unbiased regarding the subject, since object oriented examples for novices has been the main focus of my research for the last three years. Being aware of this, I have made an effort to set aside my preconceptions of object oriented quality.

The question of validity in qualitative research is a matter of standards to be upheld as ideals (Whittemore et al., 2001).

In this study the classification of statements has been validated by a test-test procedure with a second researchers coding the same data with only minor, and insignificant, differences. About 17% of all statements were randomly selected

and classified. The major part of differences in classifications, was due to the interpretation of the aspects of the theme *Examples*, which is not part of the study presented here.

By supplying a rich amount of quotations the results are transparent and allow for evaluation on credibility and authenticity. The research process has been tested and the author's long experience and training in counseling skills, working for many years as a student counselor, makes it plausible that the author is concerned of giving voice to all participants, and is sensitive to differences among participants. Therefore the criteria for credibility and authenticity can be regarded as fulfilled.

The design of the study is conscious and articulated. Furthermore, the operationalisation of the research area is structured, data collection decisions are presented, and verbatim transcriptions are provided, which implies thoroughness.

8 Summary

Several of the interviews reflect that it is a language being taught, rather than a paradigm. This can not be criticised per se, but for all ten interviewees the task given, according to syllabi, was to teach object oriented programming, or object orientation. If the mission is to teach object oriented programming, we can not restrict ourselves to teaching the syntax of a language and do it in an basically imperative fashion.

If object orientation is to be taught properly, as a problem solving approach with a distinct focus on proper objects, it is my belief that some discussion of object oriented analysis and design need to be present. Introducing a strategy or method for choosing objects in a proper fashion, to be able to make reasonable abstractions, modeling entities in the problem domain, is absolutely necessary.

> *The purpose of empirical research is not only to observe behaviour, but to* think about *behaviour. Empirical science in young domains such as CS education is not so much a process of getting answers as one of finding even better questions.* (Fincher and Petre, 2004, p.23)

Bibliography

Clancey, M. (2004). *Misconceptions and Attitudes that Infere with Learning to Program*, pages 85–100. Taylor & Francis.

Dale, N. (2005). Content and emphasis in CS1. *ACM SIGCSE Bulletin*, 37(4):69–73.

Du Bois, B., Demeyer, S., Verelst, J., and Temmerman, T. M. M. (2006). Does god class decomposition affect comprehensibility? In Kokol, P., editor, *SE 2006 International Multi-Conference on Software Engineering*, pages 346–355. IASTED.

Eckerdal, A., Thuné, M., and Berglund, A. (2005). What does it take to learn 'programming thinking'? In *ICER '05: Proceedings of the first international workshop on Computing education research*, pages 135–142, New York, NY, USA. ACM.

Elo, S. and Kyngas, H. (2008). The qualitative content analysis process. *Journal of Advanced Nursing*, 62(1):107–115.

Fincher, S. and Petre, M. (2004). *Computer science education research*. Taylor & Francis, London.

Forman, J. and Damschroder, L. (2007). Qualitative content analysis. *Advances in Bioethics*, 11:39–62.

Graneheim, U. H. and Lundman, B. (2004). Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness. *Nurse Education Today*, 24(2):105 – 112.

Guzdial, M. (2008). Paving the way for computational thinking. *Commun. ACM*, 51(8):25–27.

Holland, S., Griffiths, R., and Woodman, M. (1997). Avoiding object misconceptions. In *Proceedings of the 28th Technical Symposium on Computer Science Education*, pages 131–134.

Hsieh, H.-F. and Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Qualitative Health Research*, 15(9):1277–1288.

Kaczmarczyk, L. C., Petrick, E. R., East, J. P., and Herman, G. L. (2010). Identifying student misconceptions of programming. In *SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education*, pages 107–111, New York, NY, USA. ACM.

Morse, J. M. (1991). Approaches to qualitative-quantitative methodological triangulation. *Nursing research*, 40(2):120–123.

Nordström, M. (2010). Educators' views on object orientation. *(to be submitted)*.

Sandelowski, M. (1995). Sample size in qualitative research. *Research in Nursing & Health*, 18(2):179–183.

Shank, G. and Villella, O. (2004). Building on new foundations: Core principles and new directions for qualitative research. *The Journal of Educational Research*, 98(1):46–55.

Skolverket (2010a). The swedish national agency for education—-homepage. `http://www.skolverket.se/sb/d/353` Last visited: 2010-09-30.

Skolverket (2010b). The swedish national agency for education: Syllabuses. `http://www3.skolverket.se/ki03/front.aspx?sprak=EN` Last visited: 2010-09-30.

Thompson, E. (2008). *How do they understand? Practitioner perceptions of an object-oriented program*. PhD thesis, Massey University, Palmerston North, New Zealand.

Transcriva. Transcriva homepage. `http://www.bartastechnologies.com/products/transcriva/`.

Whittemore, R., Chase, S. K., and Mandle, C. L. (2001). Validity in Qualitative Research. *Qualitative Health Research*, 11(4):522–537.