Computer Science Education
Vol. 00, No. 00, ?? 2010, 1–16

RESEARCH ARTICLE

Educators' views on Object orientation

Marie Nordström*

Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden

(*v1.0 released november 2010*)

Much of the research on the teaching of object orientation has been focused on the students and their learning. Less is known of how the educators themselves think about different issues of the paradigm. The personal view of the educator is an important aspect that will affect how object orientation is taught. To investigate this, a qualitative study on educators' views on object orientation has been conducted and categories of views concerning object orientation, objects, and examples for object orientation defined. In all, ten educators have been interviewed, six teaching in upper secondary school and four teaching at university-level. The results indicate that educators have a simple conceptual model of object orientation, which is likely to affect the presentation of the paradigm.

Keywords: Educators, Novices, Object orientation

1 Introduction

An important aspect of teaching introductory object oriented programming to novices, is to convey the general ideas of the paradigm. Since the presentation of the paradigm is likely to be based upon the personal views of the educator, it is important to know how educators think about the paradigm. What do we know about the way educators themselves think about object orientation? What conceptual models are the basis for their teaching? There are, to our knowledge, no studies on this perspective of teaching object orientation to novices. This lack of previous work in teachers' views on object orientation makes this study exploratory in nature. If we want to discuss the teaching and learning of object orientation, we need to listen to the teachers to try to identify the basis for their approach to teaching object orientation to novices.

Discerning the 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.

To be able to listen to educators talking about the teaching of object orientation in their own words, we decided to use a qualitative approach with semi-structured interviews.

There are no prerequisites in programming for entering a CS-majors program at university-level in Sweden. Still it is often the case that students taking the introductory programming course, have previous formal training from upper secondary school. This makes it interesting to investigate the views of both upper secondary school teachers, as well as lecturers at the university-level. Due to the swedish educational system the learning outcomes of upper secondary school courses is well known, see Appendix A for details.

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

*Email: marie@cs.umu.se

ISSN: 0899-3408 print/ISSN 1744-5175 online
© 2010 Taylor & Francis
DOI: 10.1080/0899340YYxxxxxxxx
http://www.informaworld.com

2 Related Work

It has been argued that object orientation is a natural way for problem solving. However, several studies question this claim,see a survey of studies in (Guzdial, 2008). For example, 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 are major topics mentioned. Seemingly minor topics reported was: 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 respondents frequently mentioned a struggle to find a balance between object orientation and more general programming constructs. Object oriented analysis and design was considered hard to exemplify because of the, by necessity, small examples.

Most of the focus on educators has been to investigate what they think of their students' difficulties and different to teach object orientation (Clancey, 2004; Holland et al., 1997; Kaczmarczyk et al., 2010; Eckerdal & Thuné, 2005). Thompson (2008) has been exploring educators' perceptions of object orientated programs, as a result formulated a number of critical aspects and then evaluated to what extent common text books address these critical aspects. Educators, at different educational level, have been asked about what topics should be taught in introductory programing courses (Schulte & Bennedsen, 2006). A thorough analysis of the educators' ranking of topic-relevance in relation to how they ranked the teaching-level of topics according to Bloom's taxonomy, supports the conclusion that focus is mainly on coding (syntactical issues). This matches research findings that students do not gain a general conceptual understanding of programming during CS1 (Guzdial, 1995; Milne & Rowe, 2002; Lahtinen et al., 2005). Teachers' experiences of their students' successes and failures, show that there is a disempowering view among some educators that their teaching has little, if any, impact on the learning outcome of the students (Pears et al., 2007). This is contradicted by Winslow (1996), who argues that educators must supply models to the students. Models of control, data structures and data representation, program design, and problem domain are all important. Models are crucial to building understanding, and if the instructor omits them, the students will make up their own models of dubious quality.

3 Method

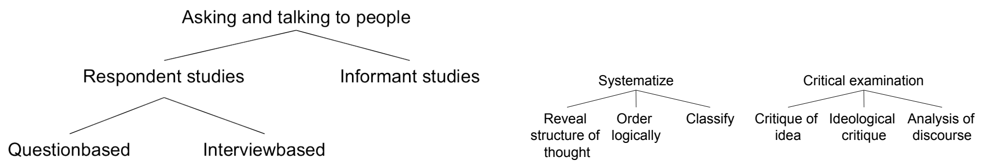
The research question was thematically operationalised according to four themes; the paradigm itself, the concept of an object, examples, and object orientation analysis and design. Each theme was viewed from three different aspects, the educator's personal view, the educator's view of student difficulties and the educator's choice of methodology to meet those issues, illustrated by the matrix in Figure 1. The reason for choosing this structure was an attempt to separate the different components concerning the design of examples. The way examples are chosen or designed is probably affected by the preferences of the educator.

The goal of all qualitative research is to understand a phenomenon. According to the definition by Esaiasson et al. (2007) there are two types of studies when talking to people; respondent studies and informant studies, see Figure 2(a).

In this classification informants are witnesses and it is the different stories of a certain event that aids the researcher in putting the picture together. In these cases there is no need to ask the same questions of all the informants. In a respondent study it is the individuals themselves and their thoughts that are the object of study. In this case it is the aim of the research to find common patterns and themes in the stories told by the respondents and it is therefore important to ask or discuss the same questions.

	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?

Figure 1. Interview guide.



(a) Types of studies (b) Reseach Questions for Qualitative Studies

Figure 2. Types of studies and research questions in qualitative research.

Qualitative research can answer different kinds of questions. Figure 2(b) shows an overview of the aims of content analysis. The reason for choosing this structure was an attempt to separate the different components concerning the design of examples. The way examples are chosen or designed is probably affected by the preferences of the educator.

The purpose of the study presented in this paper, was to explore and investigate ways of thinking about object orientation. Shank & Villella (2004) use the metaphor of a lantern to describe qualitative research. This kind of research sheds light on areas previously obscured.

3.1 *Sample*

In all, 10 interviews were conducted, 6 with educators 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. The group of interviewees has been recruited through mail-contact based on either web-based searches or recommendations from university colleagues, some through a net-work for programming educators in upper secondary school. Many potential schools did not teach object orientation at all. Contact was made with responsible director of studies or similar, and if the school did teach object orientation, the request was either forwarded directly by them or a name was given to me. The final set of respondents is therefore a convenience sample. Nonetheless, the population shows a diverse background, see the demographics in Figure 3.

The columns in Figure 3 describe:

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

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

Figure 3. Demographics of Interviewees.

Systems. T* is on his/her way to a teachers degree, but not graduated at the time of the interview.

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

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 [fig:IPdata] shows the respondents' experience, in years, in teaching programming (Exp).

It was not easy to find any 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), based on work by Morse (2000).

All of the upper secondary school educators (id R1-R6) are trained teachers in maths and/or physics. Another common background 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 teachers background in upper secondary school is probably due to the fact that Computer Science is not recognised as a subject within the teacher education system. The lack of trained CS-teachers makes it common for schools to assign science teachers, without formal CS training, to teach programming courses. Teachers in upper secondary schools are often autodidacts and on many schools they are also the only teacher teaching this subject. In is not uncommon in computer science departments for university educators to teach before and during their PhD-studies. An interesting note is that one of the university lecturers in this study earned a PhD in Chemistry before switching to CS.

3.2 *Interviews*

All interviews were conducted at a place chosen by the interviewee. The interviews were recorded using a digital voice recorder, and the length of the interviews ranges from 45 minutes to 1 hour and 16 minutes. All the interviews were conducted by the author and in Swedish. Every interview started with the interviewer asking the interviewee to describe his/her background and how he/she came to be teaching object orientation to novices.

The transcription was done verbatim using the program Transcriva (????). Some of the interviews were transcribed by the author, and for the remaining interviews, the transcription was directly supervised by the author. The transcripts were all proofread by the author, and any discrepancies, unsolved obscurities or misinterpretations, corrected by the author. Finally, all quotes shown in this paper have been translated by the author.

In an effort to limit the effect the interviewer might have on the interviewee, the vocabulary was kept at a non-formal level, not to influence or intimidate the interviewee unnecessarily.

3.3 *Analysis*

The analysis has been done using qualitative content analysis (Hsieh & Shannon, 2005; Forman & Damschroder, 2007). Content analysis is a widely used qualitative research technique, particularly in health studies (Graneheim & Lundman, 2004; Hsieh & Shannon, 2005; Forman & Damschroder, 2007; Elo & 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 and, hence, adhere to the naturalistic paradigm. 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 & Shannon, 2005).

In this study the conventional approach has been used, because of the lack of previous studies on educators' views on object orientation. The primary objective of the study, is the manifest view of object orientation. This is investigated through the different aspects presented in Figure 1.

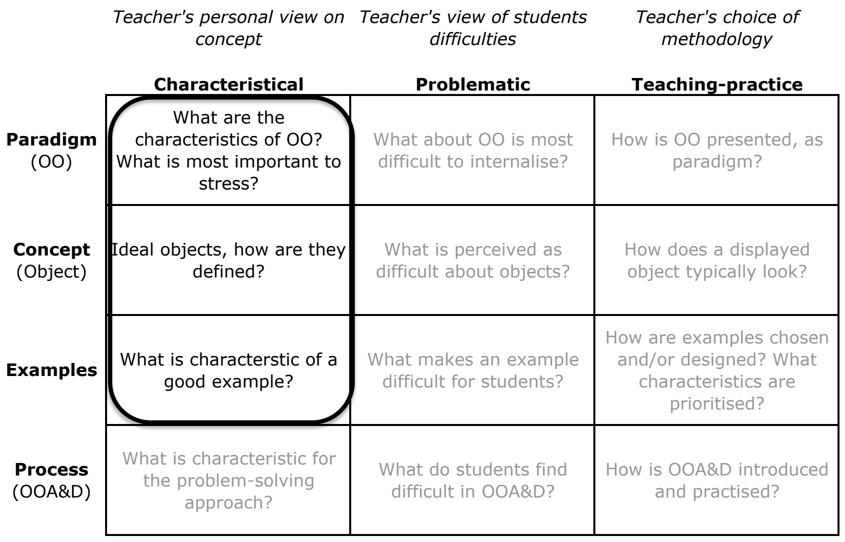
Once the transcripts were done and proof read, each was transferred from a word-document to a spreadsheet-document. All statements in the transcripts have been given an identification code [id_row]where id is the respondents identification, seen in Figure 3, and row is the row number of that particular transcript. Reading through the text, interesting statements were condensed/concentrated, in a separate column. Next, 13 columns were added, the first twelve for marking any of the 12 aspects in Figure 1, and the last column to mark other interesting comments in the text. Then the interviews were processed over again and each concentrate was labeled as belonging to one or more of the 13 aspects. After filtering out all tags belonging to a certain aspect, e.g. Educators personal view on object orientation as paradigm, in all the interviews, the next step was to observe any patterns or themes among them. According to Forman & Damschroder (2007) the coding allows the data to be rearranged in analytically meaningful categories. The selected concentrates were organised into thematic categories, sometimes in several passes, to achieve a suitable level of abstraction.

During the analysis both the audio files and the transcripts have been processed many times, and, if necessary, corrections of the transcripts have been made throughout the work.

4 Results

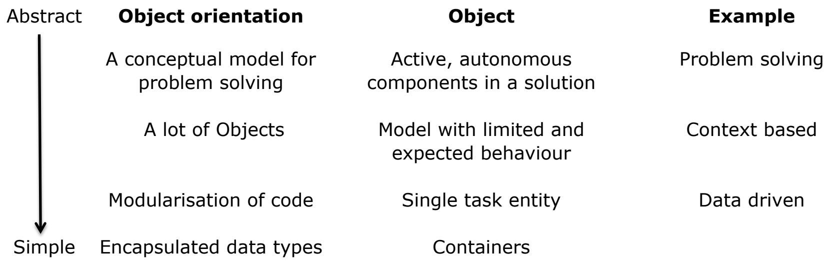
Based on the process described in Section 3.3, three aspects from Figure 1 were analysed: *Educators personal view on the characteristics of object orientation*, *Educators personal view on the concept of objects*, and *Educators personal view on examples*, se Figure 4.

The resulting categories are shown i Figure 5.



	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?

Figure 4. Aspects analysed.



	Object orientation	**Object**	**Example**
Abstract			
	A conceptual model for problem solving	Active, autonomous components in a solution	Problem solving
	A lot of Objects	Model with limited and expected behaviour	Context based
	Modularisation of code	Single task entity	Data driven
Simple	Encapsulated data types	Containers	

Figure 5. Categories.

4.1 *Object orientation*

Asking the educators for their personal view on object orientation was more complicated than expected. Some of the respondents tended to give examples rather than to formulate some theoretical or conceptual point of view. Often their personal view had to be collected from statements discussing more practical issues of their teaching.

Four categories emerged from the analysis. Object orientation is characterised as: *A conceptual model for problem solving*, *A lot of Objects*, *Modularisation of code*, or *Encapsulated data types.*

A conceptual model for problem solving

On the most abstract level of description R7 phrases it like this:

> **I :** Thinking more generally about OO, what is your personal view, how do you think of OO, what is kind of the central...
>
> **R7_143:** [...] I like... or what I find appealing about it is on one hand that it is a way of viewing not only computer programs but also activities and... phenomenon that you would like to describe. So you have a, foc.. some kind of spectacles or raster that you could apply when regarding something.
>
> **R7_146:** Then using the same raster when making a program that in a way is related to this activity, so that this is a support to address two different types... of understanding of activities and the implementation and the design of computer programs. I find it a real strength to have a common language [for these two aspects of work, authors note]

Collaboration is rarely mentioned, but is often implicitly present. R8 is one of the few stating it explicitly:

> **R8_197:** [. . .] the main idea is that I want to present classes and objects and try to make small examples where objects collaborate to solve a certain task.

However the way to do it is a problem to him/her:

> **R8_197**: [. . .] Eh .. and . . . I do not think that I succeed so well with this in this course, because it is a very small part of the course, eh. . .

A lot of Objects

The idea that object orientation is characterised by a many objects, is expressed explicitly by many of the interviewees. The concept of an object is however not clear from this expression. See Section4.2 for further discussion on this topic. R4 uses the metaphor of a smrgsbord to explain object orientation:

> **R4_158:** I think that. . . ,you have to understand that object orientation is lots of objects that you use, or lots of classes that you put on a smrgsbord and make a program out of it. And they work together.

The nature of the collaboration is not entirely clear, R4 continues:

> **R4_158:** [. . .] you kind of pick, here is this object and I pick this method from this object and use it in my program, then I pick this object from here and use it in my program And then they work together. [. . .] Object orientation is that you have a method her that you call that does a lot of things for you, you don't really need to know what it really does, but all you have to know is what comes out of it.

Several of the respondents emphasises the need to be familiar with the API of Java (or similar libraries connected to other languages). R4 continues and elaborates on this:

> **R4_172:** If you master that [using classes from a library] it does not matter if it is Java, Pearl or whatever. Eh, you, you kind of see that, here are these classes, which ones do I need to make a program that does this, well then there is this method in this class and there is this method in this class and then I pick them and put them here.

> **R3_75:** if it feels familiar [when trying out a new language] and you know how a language works with API's and object orientation with classes and then it is not difficult to switch to another. . . instead of switching between lots of.. on one hand it is good to have tested a lot of languages, but then I think it becomes. . . . then they only get to try then they never get the possibility to get deeper into it I think, that is my personal view. It becomes to jumpy. . .

Modularisation of code

The use of object orientation as a way to structure code is often mentioned by the interviewees. Asked for a personal opinion on on the characteristics of object orientation R1 replies:

> **R1_188**: Ehmmm. . . Oh, that was a difficult question. Yes well, or an extensive question. . . . Well really it is, in a way, this thing about. . . . to separate. . . no, to split the problem in. . . in smaller pieces . . . that you should. . . you can view this in different ways. One way that you create your own variable types, that is. . . I usually have an exercise in the beginning, which I don't have now, but used to have where they should make one for complex numbers. A, A class for complex numbers. Ehm, which you then can use, that is one view of object orientation that you reuse code and that you. . . . package. . . all that belongs to. . . it. And then when you have the final package then it is done, like it is. . . . well. . . .

Among others, respondent R8 has to deal with the problem of a very inhomogeneous group of students. Some of the students are not taking any more programming classes, and because of them R8 builds on the procedural way of modularising a problem:

> **R8_188:** [. . .] Okay, let's take this thing with functions one step further, and then we create classes, that contain function and data then.

R8 does not consider this suitable for the CS-majors, that take the same class. When further probed for the essence of object orientation R8 continues to state that he/she wants to show collaborating objects that solve a certain task (Section 4.1 quote **R8_197**). This educator would like to use different views depending on the audience, and is in fully aware that the CS-majors suffer from being forced into the same class a a group of students taking only this particular course.

Encapsulated data types

Several of the respondents have difficulties discussing object orientation from a conceptual point of view. Encapsulation is mentioned when asked about central concepts.

> **I:** I am just going to recapitulate [. . .] You mentioned that what you think is central in object orientation is encapsulation. This thing about not fiddling with private data and. . . that, that one takes an outside view of the object.

> **R5_418:** Yes. And that, eh. . . . that they [students] have great trouble in understanding the need for this.

To try to identify R2's view, he/she was asked if there was anything he/she considered contradictory to object orientation, the answer was:

> **R2_411:** [. . . (thinking) . . .] well, but if you look at this thing of encapsulation, which I find, that is a rather central part, how you treat data internally with an interface, when you sort of depart from that you have, you ignore that thing with encapsulation and just move on. Eh. . .

Probed for if this could mean, for instance, public attributes, R2 continued:

> **R2_417:** [. . .] for instance, I read in one of the books that if something is set to public you don't have any problems, and that is kind of, that is true, but then, then you loose something as well. . . I think, because it, it is a huge point in protecting them so that they kind of. . . only can be modified in a, in a predefined way.

On the lower scale of abstraction, R9 is thinking about his/her way into object orientation:

> **R9_972:** [. . .] I found it very difficult to understand the reason for object orientation compared to procedural programming, but I have not arrived at the answer yet (laughing)

> **I:** So how do you describe this [difference] to the students?

> **R9_978:** (laughing) Luckily, they do not ask me that kind of strange questions. . . (laughing)

R9, a bit further into the interview, argues that platform independence is the main argument to make students accept to learn Java, and that object orientation in itself could not be used to motivate students.

4.2 *Objects*

The different categories describing the view of the object as a concept are: *Active, autonomous components in a solution*, *Model with limited and expected behaviour*, *Single task entity*, and *Containers*.

Active, autonomous components in a solution

Discussing the essence of object orientation, R10 comments on the difficulty moving on from an imperative approach to an object oriented one with active, participating modules:

> **R10_95:** [...] moving on to active modules, modules that can take on the responsibility for its neighbourhood, that is such a new and different way of thinking. Ehhh... and that is where I think the students get stuck and makes... tries to make passive modules instead, and then I don't think that you have taken this step [to object orientation].

R7 expresses similar ideas, he/she wants the students to see programming as a way of solving problems and states his/her view of objects:

> **R7_338:** well you know I... and this relates back to my aversion towards the usual examples that you find in Java-books or programming books and on the web and, that... I have a difficulty seeing.... I find Customer as a realistic example of something that many of them could be working on in a future profession.

R7 then reflects on the success of their efforts:

> **R7_380:** [...] I do think that we have managed to convey this way of viewing the world as consisting of objects that originates from some kind of abstract model or how to put it and yes...

But further into the interview, R7 has to admit that the students have a difficulty moving from the conceptual view to actually gaining skills in implementing their designs.

Model with limited and expected behaviour

In a slightly more limited view, some of the respondents discuss natural models. Object should be properly instantiated and behave naturally. R1 is looking at an example with a Die, and is surprised that there is no constructor that takes the number of faces as parameter:

> **R1_779:** [...] I would choose to have one more constructor

This particular Die-class has set- and get- methods for accessing the faceValue of the die. R1 finds this strange:

> **R1_779:** You can not set.... No, it ahh. That depends on what you want it for, but I don't see how you.... why you should set the value of a die-roll.

> **R1_791:** [...] well, you think about a die really, and the only thing you can do with it is to roll it and to read the face value. An that is kind of....

R4 is also discussing the limited behaviour of ideal objects:

> **R4_320:** an ideal object... an ideal object, if you consider having a very particular, if I may return to the gambling-context, an ideal object is an object that does a very specific thing that makes very specific, that can perform specific tasks. I think that you must not do too complex objects, how shall I put it? [...]

R4 then moves on to comment on the fact that the same entity can be a good candidate for a class in one context but too complex in another. This long elaboration ends with a summary of ideal objects:

> **R4_320:** [...] So to me, the objects must not be ehh, may not be too complex because then I would kind of split them into smaller pieces, because if they get too complex then it will be too many methods. It is like 586 different methods and 373 different attributes and that, no one will ever have the energy to learn that.

R9 indicates the possibility to take an outside view of objects, to think of objects the way a potential client would:

R9_423: Methods, often I compare this to consulting. I usually explain it that way. The, it's kind of a... you order a service from a consultant. Ok. And then I move on to what the consultant needs, the parameters. What the consultant needs to do my work.

This is the only occasion in all the data that anyone expresses the use of an outside view of objects.

Single task entity

Conceptually slightly less complex is the idea of small units. R3 was asked what kind of objects he/she would never show to students, and expressed his/her view like this:

R3_231: [thinking for some time] well, objects.... the classes should be properly built. The classes must be properly built, a class should do one thing, You can not mix a lot of things, I warn them about that. A class should be clean, it should perform one thing, the methods in that class should only do one thing. I am very particular about that, [...] I tell the students that, one class should be one thing, you build objects from it. Method should be... do one thing, you should not mix things in this. In that case you make a new class, in hat case with that object. Try to separate the code so that...

R4 also expresses the reasons for and consequences of simplicity:

R4_329: [...] and that is why we in Java and all these I mean, is it anyone who cares to count all the classes and objects that there is in Java, it is thousands of millions of objects in Java, and I think that this is due to the fact that you don't want objects to be too complex, but that objects in themselves should be fairly simple and that there might be other objects that inherits from this object, and altogether it becomes a huge amount of methods maybe if you think about all the methods they inherit and themselves.

Containers

Several of the respondents expresses the view of objects being mere containers.

R3_303: Well, because it is data, we look more at the data kind of that.., what will you collect data for? You look at it [the data], well then this should be an object of this data and then... well you can't mix this into this object but now this becomes another object. Now we have to build this to make it. So finally when you collect the data together you can see what you should have, the demands around that I think.

R5 has made a choice to base all his/her application around databases. Everything has to stored in some way, and the natural solution to this is a database.

R5_25: Well, we... we do have something that I think is really good when you're dealing with object orientation, that is that you have a database beneath, in some way... because they... the model-view-control thinking. Model is almost always... ehhh, a table in the database. That is not particularly.. it's almost always that way. Then we get into, what is it- then we get into something like instead of talking about objects we talk about, what is it that we need to store, what is there, are they related etc. And that is really the same as object orientation.

R5 continues to comment on the fact that this enables him/her to discuss the importance of storing information only in one spot and relationships among data.

R5_25: I feel much more comfortable with... with discussing how to normalise a database and how to design a database than discussing how you... which.. cl... objects we should have. And they are rather close, I think... in the end once you are starting to implement it.

R5 also finds graphics useful and driving object orientation:

R5_25: Those that... [for their project] make a game, that becomes o- that becomes rather object oriented. Because they have so- some character walking.. meeting a lot of zombies and then you make a zombie-class and then it is generated.... a lot of zombies that you...fire at and then every bullet is...an object too and so on, so it becomes very...but they are...that it became so object oriented depends on the fact that they write their code in Flash which is- , and there's nothing else there but objects that in addition are graphical .. eh.. objects.

No other respondent takes such an explicit stand on how to define, and how to work with objects.

4.3 *Examples*

Discussing object orientation and how to exemplify it is easier than discussing the more generic characteristics of object oriented examples. Listening to the respondents, there are however bits and pieces here and there that gives a description of their view of examples. The emerging categories are: *Problem solving*, *Context based*, and *Data driven.*

Problem solving

R7 has previously stated his/her view of object orientation as a problem solving tool, and this is also reflected in his view of examples:

R7_344: [moving on from the example of `Customer` **R1_338**] but, `Car` and `Bike` and things like that, ok it might work from a pure programming point of view, but I think that our students, if you generalize, I am not going to do that, but many of our students don't view programming as...as this, this pleasurable, self-sufficient activity. But they see it as a means for something else. And if I don't offer examples that...makes it credible that programming is a tool that they might actually need to do this, then I won't get through to them. They just don't see the use of being able to represent Cars in a computer program.

R2 is also expressing a view of of object orientation being so much more than the technical part of programming, and that this has to show in examples:

R2_246: Ehh, I try to find examples that are as pedagogical and close to reality as possible, I think that, a shortage in some of the course literature is that it gets too technical, that is, they just show the technical details of programming and not kind of, connects to the problem, because I think that object orientation is much more about problem solving than programming. In...that job is, later when you're done, then it becomes pure programming and that is something...something slightly different, that is another skill.

Most of the respondent are at several occasions commenting on the lack, or shortage of, really good examples. Despite this complaint it was almost impossible to have them characterize a good example.

Context based

The need for examples to be contextually situated is mentioned by many of the respondents explicitly:

R7_287: so what I'm looking for and try to design myself, that's classes that, if I try to make a connection to the course they have taken before programming, when they have been looking at activities, then I try to use classes of the kind `Customer`, ... and ... well, things that are relevant or how to put it. Something that you, that seem reasonable to.. [disrupted by a person entering the office]

R9 is very concerned of making the example connect to something in the every-day life of his/her students:

R9_204: It is ... now I only have students from X [exchange students] , then it is kind of easier to explain to them. Ok, you have arrived here to study and your parents want to transfer money to you. Do you want them to transfer the money to my account or yours? Then they want them to transfer the money to them because... Then you have to go to the bank an get an account. And then I explain to them that when you go to the bank if you... well, creates an account what does that mean? It means you have to have a social security number. You want to have an address and name and so forth. Then you create an object, each person that you create an account for becomes an object of the class `BankAccount`. Or something like that...

The context of bank and bank accounts is a common theme in textbooks, and is mentioned by R3 as well

R3_171: Where you deposit and withdraw money. Because I got that from.. I think I got that from U [mentions the university where he/she received his/her degree]. That it is... `BankAccount` is the name of the class, and there you have only name and you have a balance. Two variables. String name, and int balance. And you have `getBalance` and `setBalance` and a constructor where you can supply values, the name and the lance that you have to begin with. So I think it is... they understand this sufficiently, a bank, because they understand this, deposit money, withdraw money from an account it's only two variables which isn't much, that is what I start out with.

In upper secondary school, all the teachers have to take the students from procedural to object oriented programming. R4 tries to exemplify this difference, and describes how he/she chooses his examples:

R4_182-266: Eh, that, that I take, I take an example close to them [students], [...] Ehm, and that in object orientation you explain, with the help of objects, a world. That, for instance, that we would like to describe a forest in procedural programming them you see things that might happen in the forest, functions, the sounds of birds plants leafs falling etc. In object oriented programming you see the objects that the forest is made up of, trees, rocks, birds, and so on, that, I try to, something like that, that was the forest before and this is the forest now.

Data driven

For many of the respondents, typical examples are focused on data to be stored. R5 can not think of any other origin for his/her presentation of a problem:

R5_25: Then we get into, what is it- then we get into something like instead of talking about objects we talk about, what is it that we need to store, what is there, are they related etc.

Some of the respondents use the term data type in connection to classes, generally meaning something like a record in Pascal:

R2_309: Well then, then it's more of a, a, well you, you have a need for a certain data type, a composed data type, eh, it might be a ... what later on would be a post in a database or something like that, or ... eh.... a country with, with certain data that belongs or, a document med certain characteristics. Eh.... where you collect, eh attributes and methods in a class and by this showing that it kind of belongs together.

5 Discussion

The purpose of this study has been to look for different ways of viewing object orientation, not to categorise individual educators. Several of the respondents discuss object orientation using several views, and some only express a single view.

An interesting observation is that, even though 8 of the 10 educators have received formal university training in object orientation, the level of abstraction in their views of object orientation is low. Although many of the upper secondary school educators have formal training in terms of university courses, this is usually not included in their teachers training program. Only one of them has moved on from a CS-major degree to complement with a teaching degree. But as apparent from the quotations, a university degree does not in itself guarantee that the view of object orientation is in line with generally accepted concept definitions. One interesting observation is the choices of application areas. Only two of the respondents explicitly want their examples and contexts to be realistic, in terms of software. Other respondents, who take on a more data driven design approach, settle for applications where it is obvious that a large amount of data should be handled. The storage of data then becomes the driving force for object definition.

Students' conceptions of object orientation after attending a CS1 course have been investigated through concept maps, by Sanders et al. (2008). CS1 is by necessity focusing on both introduction to programming as well as introduction to the object oriented paradigm, which shows in the students' concept maps. Few maps indicate any relationships other than among the syntactical components of programming. This corroborates the resulting categories of our study. For the majority of the respondents, the views of object orientation were displaying a focus on programming/syntax skills, rather than emphasising a conceptual view.

In the present study there is no single mentioning of abstractions, and behaviour is only discussed in terms of methods, or even functions, instead of, for example, responsibilities. A majority of the educators view classes as primarily data types, consisting of primitive values to be manipulated. This does not promote an outside, problem-oriented perspective with collaborating objects as autonomous service providers for clients. Only one of the respondents indicates this view, using the metaphor of consultants to describe methods to students.

The categories reflecting the educators views of object orientation, object and examples are related on a conceptual level and are ranging from elementary syntax-based views to abstract, problem solving views in the three aspects investigated, *Educators personal view on the characteristics of object orientation*, *Educators personal view on the concept of objects*, and *Educators personal view on examples.*

6 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 tactical in their description of certain subjects.

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. The quality of examples in text books must be considered low, as evaluated in previous work (Börstler et al., 2008, 2009, 2010), and this raises questions regarding the view of object orientation among educators in general. Being aware of this, I have made an effort to set aside my preconceptions of object oriented quality, when analysing the data.

7 Summary and Conclusions

In this study, we have investigated how educators describe their personal views on object orientation. This has been done through qualitative analysis of ten interviews with educators from upper secondary schools and universities.

It seems that educators's views are more focused on the handling of data, rather than the object orientated approach to problem solving. This makes objects mainly containers of data with set- and get-methods to manipulate them. The few examples that are chosen to show the idea of object orientation are not related to a software problem, but rather models of natural environments, for example "The Forrest". They do not have any counterpart in examples for specific object oriented concepts or features.

The way educators view object orientation and objects will have a large impact on students being introduced to the paradigm. Since only two of the educators, both university lecturers, expressed thoughts about an explicit aim to teach the foundations of object orientation, it is important to address this lack of presentation of the paradigm. There is no doubt, that educators always have to compromise with principles for practical reasons in their teaching. Audience, time, and space will restrict the possibilities to show novices object orientation at its best. Nevertheless, it must be the ambition of educators to convey the essence of the object oriented paradigm even in details. There is a lot of work to be done in terms of practical suggestions for line of presentation and good examples to support the struggling educators.

Acknowledgement

The author is greatly indebted to the participating educators. Without their willingness to share their thoughts and experiences, and to devote time and effort to this project, this research would not have been possible.

References

Börstler, J., Christensen, H.B., Bennedsen, J., Nordström, M., Kallin Westin, L., Jan-ErikMoström, et al. (2008). Evaluating OO example programs for CS1. In ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education, Madrid, Spain (pp. 47–52). New York, NY, USA: ACM.

Börstler, J., & Hadar, I. (2008). Eleventh Workshop on Pedagogies and Tools for the Teaching and Learning of Object Oriented Concepts. In ECOOP 2007 Workshop Reader, Vol. LNCS 4906 of *Lecture Notes in Computer Science* (pp. 182–192). Springer.

Börstler, J., Hall, M.S., Nordström, M., Paterson, J.H., Sanders, K., Schulte, C., et al. (2009). An Evaluation of Object Oriented Example Programs in Introductory Programming Textbooks. *Inroads*, *41*, 126–143.

Börstler, J., Nordström, M., & Paterson, J.H. (2010). On the Quality of Examples in Introductory Java Textbooks. *The ACM Transactions on Computing Education (TOCE)*, *Accepted for publication.*

Clancey, M. (2004). In S. Fincher & M. Petre (Eds.), *Misconceptions and Attitudes that Infere with Learning to Program.* (pp. 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., & Temmerman, T.M.M. (2006). Does God Class Decomposition Affect Comprehensibility?. In P. Kokol (Ed.), SE 2006 International Multi-Conference on Software Engineering (pp. 346–355).

Eckerdal, A., & Thuné, M. (2005). Novice Java programmers' conceptions of "object" and "class", and variation theory. In ITiCSE '05: Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, Caparica, Portugal, July (pp. 89–93). New York, NY, USA: ACM.

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

Esaiasson, P., Gilljam, M., Oscarsson, H., & Wängnerud, L. (2007). *Metodpraktikan–Konsten att studera samhälle, individ och marknad (In Swedish).* Norstedts Juridik.

Forman, J., & Damschroder, L. (2007). Qualitative Content Analysis. *Advances in Bioethics*, *11*, 39–62.

Graneheim, U.H., & 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. (1995). Centralized Mindset: A Student Problem with Object-Oriented Programming. In Proceedings of the 26th Technical Symposium on Computer Science Education (pp. 182–185).

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

Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding Object Misconceptions. In Proceedings of the 28th Technical Symposium on Computer Science Education (pp. 131–134).

Hsieh, H.F., & 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., & Herman, G.L. (2010). Identifying student misconceptions of programming. In SIGCSE '10: Proceedings of the 41st ACM technical symposium on Computer science education, Milwaukee, Wisconsin, USA (pp. 107–111). New York, NY, USA: ACM.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. (2005). A Study of the Difficulties of Novice Programmers. In Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (pp. 14–18).

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, *7*(1), 55–66.

Morse, J.M. (2000). Determining Sample Size. *Qualitative Health Research*, *10*(1), 2–3.

Nordström, M. (2009). , He[d]uristics – Heuristics for designing object oriented examples for novices. Licenciate Thesis, Umeå University, Sweden.

Pears, A., Berglund, A., Eckerdal, A., East, P., Kinnunen, P., Malmi, L., et al. (2007). What's the problem? Teachers' experience of student learning successes and failures. In R. Lister & Simon (Eds.), Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Vol. 88 of *CRPIT* (pp. 207–211). Koli National Park, Finland: ACS.

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

Schulte, C., & Bennedsen, J. (2006). What do teachers teach in introductory programming?. In ICER '06: Proceedings of the second international workshop on Computing education research, Canterbury, United Kingdom (pp. 17–28). New York, NY, USA: ACM.

Shank, G., & 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.* Massey University, Palmerston North, New Zealand.

Transcriva (????). , Transcriva Homepage. `http://www.bartastechnologies.com/products/transcriva/`.

Winslow, L.E. (1996). Programming pedagogy—a psychological overview. *ACM SIGCSE Bulletin*, *28*(3), 17–22.

Appendix A: Programing Education in Sweden

The Swedish National Agency for Education (*Skolverket*) (Skolverket, 2010a) is the central administrative authority for the Swedish public school system for children, young people and adults, as well as for preschool activities and child care for school children. Government and Parliament specify goals and guidelines for preschool and school. Because of this it is well known what the syllabi and requirements for programming courses in upper secondary school are ((Skolverket, 2010b). The Swedish upper secondary school is entered at the age of 16, and consists of three year programs. Based on interest the students make a choice among a number of programs with different focus, yielding different eligibility for moving on to the university level. All courses concerning computers and programming are organized in a subject called Computer technology. In Figure A1 the structure and relationships among the programming courses in upper secondary school is shown.

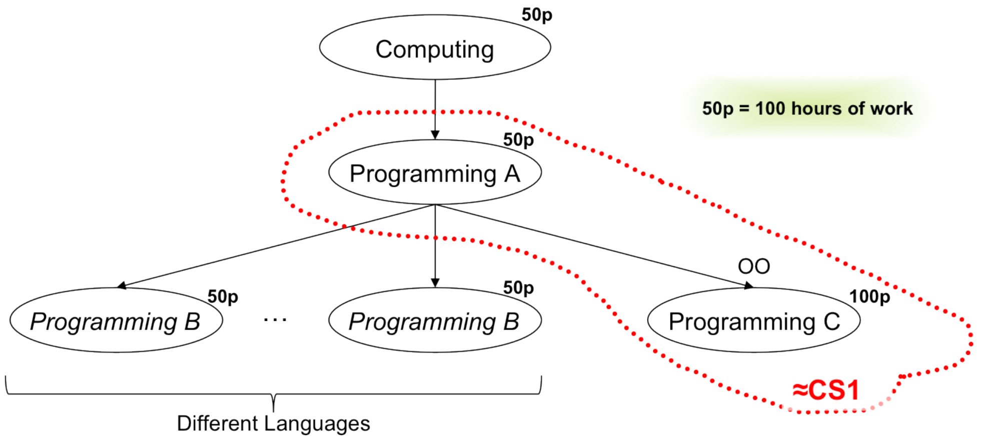


Figure A1. Computing courses in upper secondary school.

Computing is a course common to most of the programs. It 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 should provide 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. According to the syllabi, Programming A and Programming C together roughly contains the amount of stuff and time allocated to a university-level CS1 course. See Skolverket (2010b) for more details.

In Sweden, the Government has the overriding responsibility for higher education and research. It enacts the legislation and establishes the targets, guidelines and funding for the sector. At the university level the Swedish educational system is now adjusted to the Bologna system with 3-year bachelor degrees (*Kandidatexamen*) and 2-year Master degrees. In addition to this, the Master of Science in Engineering (*Civilingenjörsexamen*) is a 5-year Masters degree. These degrees are given for a number of different majors, including Computer Science. In general, the computing curricula of these programs contains traditional CS1 and CS2 courses, for both CS majors and minors.