Preface/background

Description

Object-oriented programming has been around for more than 50 years and is now the dominant style of programming. Most programming languages support object-orientation, and a substantial number of applications have been made in these languages through the years.

In recent years there has been an increasing criticism of object-oriented programming – some people advocate functional programming as an alternative and some people argue that the dominant object-oriented languages do not capture the intention of object-orientation as intended by its founding fathers, including Dahl & Nygaard [1] and Alan Kay [2] – the creators of SIMULA and Smalltalk, respectively.

Most of the criticism appears in blogs, and most of the posts do not seem to be based on solid knowledge, and they often contain several factual errors. In addition, object-orientation is also blamed for things, e.g. spaghetti code, that is obviously not grounded in object-orientation.

We do, also, hear critique of object-oriented programming from colleagues, programmers and others and there are of course valid points of criticism that gives rise to reflection.

One of the main problems with how object-oriented programming is practiced is that most authors of textbooks research papers on object-oriented programming consider reuse of code to be the main benefit of object-oriented programming.

With a background in the Scandinavian tradition of object-oriented programming with SIMULA and Beta, we think that the main benefit of object-oriented programming is oriented programming languages provide excellent mechanism for modeling phenomena and concepts from a given application domain. Reuse of code is a side effect.

We thus wrote an essay for Onward! 2022 where we discuss what we consider to be critical issues of object-oriented programming and argue that modeling should be the main focus of object-oriented programming.:

- Ole Lehrmann Madsen, Birger Møller-Pedersen:
- *What Object-Oriented Programming Was Supposed to Be: Two Grumpy Old Guys' Take on Object-Oriented Programming*.
- Onward! 2022 – Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, co-located with SPLASH 2022: Systems, Programming, Languages, and Applications,Software for Humanity.
- Association for Computing Machinery, p. 220–239.

Note! The text in this Section is mainly taken from the above essay. Also the current references are to referees in the essay.

It all started with SIMULA as a language, which was intended for modeling as well as programming. It introduced objects, classes, subclasses, and virtual methods. Classes were meant for the representations of domain concepts, and subclasses as representations of specialized concepts. A side effect was that subclasses inherited the code of the superclass.

The modeling approach of SIMULA was not dominant in most of the programming languages that followed. Instead reuse of classes was considered the main advantage of object-oriented programming. The mainstream interpretation, represented by the use of today´s dominant languages, backed by textbooks and various papers including research papers on object-oriented programming, has focus on the inheritance of code. Some languages consider classes as object factories, and subclass/inheritance as implementation inheritance: reuse of code by defining a subclass and inherit the code from the superclass and redefine what needs to be redefined. This may lead to class libraries that are difficult to understand since there is no conceptual relation between a class and its possible subclasses, in the sense that a subclass does not represent a specialization of the superclass.

Some of the blog posts claim that the current mainstream object-oriented languages do not correspond to the intentions of object-orientation as formulated by Alan Kay as part of the development of Smalltalk – according to Kay, the central idea of

object-oriented programming is that a program execution is a set of objects sending messages to each other, i.e. a message passing interpretation.

To complete the picture, there is also an interpretation that does without classes and subclasses, i.e. only have objects (and delegation). This is a prototype-based interpretation, with Self [3] as the most striking language.

Object-oriented modeling methods, e.g. [4-6], that lead to UML [7] obviously followed the original modeling interpretation. However, as stated in [8, 9], the community on object-orientation is split into a programming community that 'make programs', but do not consider modeling, and a modeling community focusing on models more or less independent of programs. We learn from programmers that they do not model as also pointed out by Bran Selic [10], and prominent researchers like W. Cook in Peak Objects, [11] do not seem to buy into the modeling interpretation.

In the above-mentioned essay, we argued that most of the criticisms of object-orientation may be due to the mainstream interpretation of object-orientation. In this interpretation, a class is just considered a factory for generating objects and inheritance is considered a mechanism for reuse.

We did argue that a program always describes a model of the application domain even if the programmers claim they do not model. But being explicit about that programming is modeling will improve the readability of software, we will argue that most of the problems with the mainstream interpretation do not exist in the modeling interpretation of object-oriented programming.

An important part of the development of the Beta language was to develop a conceptual framework for object-oriented modeling and programming – Beta book chp. 17. This conceptual framework describes conceptual means of understand and organizing knowledge about a given application domain.

However, neither the OO modeling community nor the OO programming community are explicit about such a conceptual framework. We think you need a conceptual framework to apply modeling. We restated this viewpoint in a keynote at the SAM 2023 conference:

- Ole Lehrmann Madsen, Birger Møller-Pedersen:
- What your mother forgot to tell you about modeling – and programming,
- 2023 A*CM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C) – SAM 2023.* IEEE, p. 200-210.

As a continuation of the Onward essay and the SAM 2023 keynote, and the lack of textbooks advocating modeling, we decided to embark on writing this book on *object-oriented programming as modeling*.