Types of Assignments for Novice Programmers

Alexander Ruf, Marc Berges, Peter Hubwieser

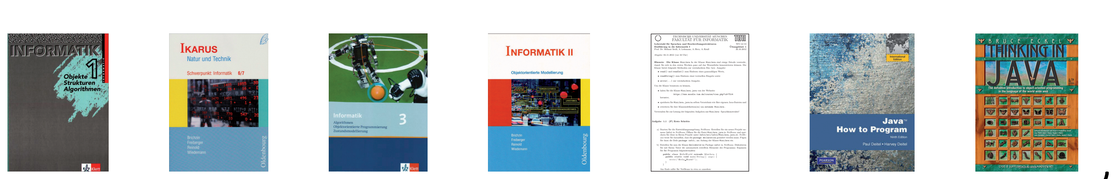
Technische Universität München, TUM School of Education

Abstract

This poster deals with the classification of assignments according to their type. In contrast to other publications, we derive assignment types not deductively, but extract them empirically from different sources. Our main research question is: What types of programming assignments are actually given to novice programmers? In addition, we compare our empirically found assignment types to the deductively derived ones from the literature. This is driven by the following research questions: Are there types of assignments that are mentioned in literature, which however are not or rarely found in actual assignments given to novice programmers? Can assignment types be found that cannot or only poorly be matched with the category types described in the literature?

Methodology

We included in our analysis all assignments of the chosen sources that contain programming code either in the assignment or in the corresponding solution. The extent of the programming code does not matter and ranges from just one line of code to the full program. Since we have restricted ourselves to assignments for novice programmers, we included assignments only up to the topics inheritance and polymorphism. Often, an assignment in the sources consists of several parts. Since the partial assignments usually differ in type we have treated and examined each subtask as an own assignment in these cases. To identify the different types of assignments, we first looked at what is given in the respective assignment and what the student has to do to solve it. Then we stripped both criteria “given” and “to do” from the context of the assignment and formulated them in a generic way. Similar “givens” and “to dos” have been combined to one assignment type, i.e. two assignments are of the same type if they have basically the same given and if the same is to do. More complex assignments, which involve more than one „to do“, were divided into corresponding parts and associated with multiple types, i.e. an „atomic“ assignment was made from each to do, which was then used for further investigation. In a last step, we tried to derive a hierarchy within the found types.



1098 Assignments

Comparison

If the task types listed by Bower in [1] are transferable on programming assignments, all of his types will be found in our empirically derived list. But the reverse is not the case, some of our types cannot be transferred to his, e.g. type 1.3 or type 2.1e. The reason for this may be because on the one hand the individual types in [1] are less accurately described and they are not specifically intended for programming assignments, on the other hand Bower’s objective was not a complete types list but a taxonomy within a list.

The types list of Hazzan and Ragonis presented in [2] and [3], is much more extensive and more precisely described. From this list only two types cannot be integrated into our list: First, the type „completing a given solution“ and second the type „efficiency estimation“. That the latter is missing in our list is probably due to the fact that these assignments are made for more advanced and not for novice programmers, which we have studied. But it is in fact noteworthy that in none of our sources a „code cloze“ occurs, especially since this type of assignment would be very suitable for beginners. Conversely, almost all of our assignment types can be transferred to the list of Hazzan and Ragonis. Of course, their classification differs in some points from ours, especially as their list is not only intended for programming assignments, nevertheless a correct mapping works almost always. Only type 2.1a, where program code is to be tested on the computer, does not match with Hazzan and Ragonis. This is probably because they have not considered this form of more practical work as a „typical“ assignment.

[1] Bower, M. (2008, June). A taxonomy of task types in computing. In ACM SIGCSE Bulletin (Vol. 40, No. 3, pp. 281-285). ACM.
[2] Hazzan, O., Lapidot, T., and Ragonis, N. (2011) Guide to teaching computer science. An activity-based approach. Springer, Berlin.
[3] Ragonis, N. (2012) Type of Questions - The Case of Com- puter Science. Olympiads in Informatics, 6, pp. 115-132.

Given	Textual description						Program code						Diagram	Prerequisites
Additionally given	Nothing	Prerequisites	Solution to a similar problem or to a part of the problem	Solution to the problem			Nothing					Prerequisites	Nothing	Nothing
To do	Write a program (or a part of it)	Write a program (or a part of it) considering the given prerequisites	Adjust or extend the given solution to the problem	Decide if the given solution is correct; give reasons for it or correct the solution	Set the right preconditions to the given solution	Optimize the given solution	Transfer the given program code to your IDE and test it	Consider the effects of executing the given code	Draw a diagram to the given code	Transform the given code into a different programming language	Consider an appropriate problem to the given code	Transform the given code according to the given prerequisites	Write a program (or a part of it) to the given diagram	Consider an assignment and solve it, considering the given prerequisites
Type No.	**1.1**	**1.2**	**1.3**	**1.4a**	**1.4b**	**1.4c**	**2.1a**	**2.1b**	**2.1c**	**2.1d**	**2.1e**	**2.2**	**3**	**4**