Universität Trier

Informatik-Wissenschaften

Didaktik der Informatik
Universität Trier

GI DdI
acm In-Cooperation
WiPSCE 2014
The 9th Workshop in Primary and Secondary Computing Education

Niveau Transformations:

A Tool for Method Classification and Lesson Planning

Martin Löhnertz - University of Trier
loehnert@uni-trier.de

A novel classification scheme for problem solving scenarios and applicable variation methods based on the relations between problem complexity, students' capabilities and features of the technological platform known to the students. This systematic approach explores alternative ways of lesson planning and analysis extending classic "didactic reduction".

🟩 switch meta level/problem

🟧 classical approaches

🟨 not applicable (?)

Transformations		Initial Niveau Relations (capability vs. difficulty)			
		👫 Students ❓ Problem 💻 Computer Students can solve the problem but not implement a working program.	👫 Students & 💻 Computer ❓ Problem Students can implement a program for an understood solution.	❓ Problem 👫 Students & 💻 Computer Students cannot solve the problem and thus not implement anything.	💻 Computer ❓ Problem 👫 Students Students apply library functions which they do not fully understand.
💻	**Empowerment** Add functions to the computing system (or allow usage)	Introduce high level functions that match the students' too abstract concepts. Pedoni, M. and Bay, T.G. 2007. Vizualize and Open Up. *Informatics in Edu.*, 6.1, 153–162	🟩 Encapsulate solution in a library and use as black box. Haberman, B.; Muller, O., "Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes". FIE 2008. 38th Annual , vol., no., pp.F1C-7,F1C-12, 22-25 Oct. 2008	Provide functions that help solve the problem and discuss modeling or parameters. Szlávi, P. and Zsakó, L. 2006. Programming Versus Application. ISSEP 2006, LNCS 4226, 48-58	Modify program to visualize solution steps. Eric Fouh, Monika Akbar & Clifford A. Shaffer (2012): The Role of Visualization in Computer Science Education, Computers in the Schools, 29:1-2, 95-117
	Restriction Remove functions from the computing system (or disallow usage)	Remove irritating redundant functions if the number of options is overwhelming.	Port code to smaller embedded or older devices. Müller, J. 2009. Der selbstgebaute Abakus. LogIn 157/158, 79-83	🟨	Disallow usage of advanced functions (e.g. "sort"). Hasker, R. W. 2005. An Introductory Programming Environment for LEGO® Mind-Storms™ Robots. MICS 2005, paper 87
❓	**Complicate** Increase size Add requirements	Enlarge the input to have the students personally experience the computational problem Gasper, F. 1991. Das Weihnachtsbaumbehängungs-problem. *LogIn* 11 Heft 6, 46-49	Request scalability, modularization, error handling etc. Husch, B. 1991. Wiederverwendbare Software-Bausteine. LogIn 11 Heft 3, 51-53	Paradox known from mathematics: More general problem might have easier solution. De Villiers, M. and Garner,M. 2008. Problem solving and proving via generalization. in Learning and Teaching Mathematics, April 2008, No. 5, pp. 19-25. AMESA.	🟩 Try to make the problem unsolvable for a computer to analyze why the original version was approachable.
	Reduce "Didactic Reduction" Add assumptions	More abstract modeling. Identify exemplary instances. Baumann, R. 2011. Eingebettete Systeme verstehen Teil 1. *LogIn* 171, 49-61	🟩 Try to automate solution process for simple instances (compiler generators). A. Demaille. Making compiler construction projects relevant to core curriculums. (ITICSE'05), pages 266–270, Universidade Nova de Lisboa, Portugal, June 2005.	🟧 Didactic Reduction. Remove special cases by additional assumptions. Schlemmer, M. 2011. Informatik Fortbildung Kommunikation in Rechnernetzen. http://tinyurl.com/k5onsyl	🟧 Didactic Reduction – solve easy instances and use the program to verify students' attempts.
👫	**Enlighten** Teach new methods Give hints	Teach methods to transform descriptive solutions into code. Strecker, K. 2011. Zur Didaktik der Algorithmik. Proc. INFOS 2011, LNI P-189, Bonn 2011, 187-196	🟩 Introduce new methods on well understood example (e.g. DIRR) Houk, S. 1999(?). "Design-Implement-Redesign-Re-implement (DIRR) – Pattern" http://extras.springer.com/1999/978-3-642-63632-5/OFFLINE/PPP/PP0.HTM	🟧 Provide hints to solve the problem. Kujath, B. 2006. Ein Test- und Analyseverfahren zur Kontrastierung von Problemlöse-Prozessen informatischer Hoch- und Niedrigleister. *LNI* P-99, Bonn 2006, 49-69	Analyze the existing program. Use-Modify-Create Approach Lee, L. et al., 2011. Computational thinking for youth in practice. ACM Inroads 2.1, 32-37
	Restrain Disallow Methods Emulate Infants Enforce objects	Reduce the pupil's abilities to that of the computer by imagination or force (e.g. Blindfolds). Diethelm, I., Geiger, L. and Zündorf, A. 2005. Mit Klebe-zettel und Augenbinde durch die Objektwelt. INFOS 2005, 149-159	Learning by teaching (imagined) younger pupils. Yu-Chen, K. and Zhe-Yu, W. 2013. An online Peer-Tutoring Platform for Programming Languages based on Learning Achievement and Teaching Skill. IJITCS 7.4, 65-70	Quit searching solution and solve by brute force methods. Anany Levitin and Mary-Angela Papalaskari. 2002. Using puzzles in teaching algorithms. SIGCSE Bull. 34, 1 (February 2002), 292-296.	🟨

*derived from "Neumagener Schulrelief" by Rheinisches Landesmuseum Trier (Foto: Thomas Zühmer) [CC BY-NC-SA]

Poster available for download:

http://www.loehnertz.de/martin/didaktik-der-informatik/wipsce14-poster