Designing a Pedagogical Framework for Developing Abstraction Skills

Marjahan Begum*
City University of London
United Kingdom
marjahan.begum@city.ac.uk

Julia Crossley*
City University of London
United Kingdom
julia.crossley.2@city.ac.uk

Filip Strömbäck*
Linköping University
Sweden
filip.stromback@liu.se

Eleni Akrida
University of Durham
United Kingdom
eleni.akrida@durham.ac.uk

Isaac Alpizar-Chacon
Utrecht University
Netherlands
i.alpizarchacon@uu.nl

Abigail Evans
University of York
United Kingdom
abi.evans@york.ac.uk

Joshua B. Gross
California State University Monterey Bay
United States of America
jgross@csumb.edu

Pontus Haglund
Linköping University
Sweden
pontus.haglund@liu.se

Violetta Lonati
University of Milan
Italy
lonati@di.unimi.it

Chandrika Satyavolu
North Carolina State University
United States of America
jsatyav@ncsu.edu

Sverrir Thorgeirsson
Federal Institute of Technology
Switzerland
sverrir.thorgeirsson@inf.ethz.ch

ABSTRACT

Abstraction is a fundamental skill and concept in computer science and it is also a difficult skill to teach. The purpose of the working group is to analyse different perspectives of abstraction's conceptualisation and ways of teaching the skill. Therefore as a result of the working group we will be first identifying how abstraction is discussed and defined in key literature. As a team we will agree on the perspectives and models we will like to explore in teaching context. Finally we will work with computing educators and computing education researchers to design a pedagogical framework that will enable the development of the abstraction skills.

CCS CONCEPTS

• **Social and Professional topics** → **Computing Education**.

KEYWORDS

abstraction, pedagogy, computational skills, CS1, CS2, CS3

ACM Reference Format:

Marjahan Begum, Julia Crossley, Filip Strömbäck, Eleni Akrida, Isaac Alpizar-Chacon, Abigail Evans, Joshua B. Gross, Pontus Haglund, Violetta Lonati, Chandrika Satyavolu, and Sverrir Thorgeirsson. 2024. Designing a Pedagogical Framework for Developing Abstraction Skills. In *Proceedings of Innovation and Technology in Computer Science Education (ITiCSE '24).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

*Leader

ITiCSE '24, July 08-10, 2024, Milan, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 BACKGROUND AND RELATED WORK

Abstraction is a key concept in Computer Science Education that plays a crucial role in shaping the way students understand and interact with computational concepts. It serves several purposes, such as the ability to focus on essential information while omitting unnecessary details, or generalising disparate information to an expression that is more descriptive. Understanding how abstraction is conceptualized on an intricate scale can help inform how it is applied by students, and how it is taught and assessed. This is essential to improving learning experiences and honing the development of crucial computational skills. Broadly speaking, the ability to abstract is seen in being able to judge the necessity of information, whether by removing, adding or identifying common traits. In Computer Science this is often manifested in being able to break problems down, construct solutions up and fluidly move between levels of specificity [3, 14].

According to Piaget, the process of reflection brings together the full range of adult cognitive abilities. Reflective abstraction involves making observations, linking them, and making inferences beyond observable phenomena through logical and hypothetical deductions [12]. In turn the Constructivist approach states that learning occurs through reflection, making it essential to present content and exercises that encourage deductive reasoning for developing inferential skills [2]. This is supported by studies examining effective indicators of abstraction ability [9]: the highest level of learning is observed through creation via reflection on knowledge.

Explicit frameworks for the classification of abstraction skills exist. Perrenet et al.'s study [13] documents abstraction using Richard Skemp's model of abstraction [20]; performance is judged against thresholds at each undergraduate level. This same model is used by Hazzan for empirical purposes [16]. Hill et al.'s triad classification of abstraction as conceptual, formal, and descriptive takes inspiration from Piaget but may not capture the same type or level of abstraction [11].

Hierarchical frameworks of concept formation such as the Block model [18] and Sfard's model [1] have been applied to computational skills and mathematical skills respectively. There is potential for applying them to the teaching of abstraction skills to undergraduate CS students.

Empirical studies in Computer Science Education have explored abstraction skills, often focusing on qualitative aspects [6, 7, 15, 19]. Where focused in higher education, these studies are aimed at pulling out insights rather than providing actionable points in details for HE educators.

Frameworks in Computer Science Education

We are evaluating frameworks of the following types:

- categorizations of abstraction skills [4, 10, 11, 13];
- hierarchical frameworks of concept formation skills [1, 18];
- frameworks for effective knowledge transfer [8];
- evaluation of processes that offer the best indication of abstraction ability [9].

These are present in both Mathematics and Computing education research. We feel these are relevant in that they can inform the design of guidelines and structuring pedagogical framework that will support the development abstraction skills.

2 GOAL OF THE WORKING GROUP

The goal of this working group is to apply current research findings in abstraction to the development of a pedagogical framework [21] underpinned by constructivism [2].

Our pedagogical framework. This working group is motivated by the fact that it is difficult to teach, understand, and apply abstraction in problem solving scenarios in CS1, CS2, and CS3. As a group we are identifying, evaluating and analysing abstraction models in the literature to see if they can be translated and transferred in the context of our diverse curriculum setting. Our current definition of a pedagogical framework is:

- learning and conceptualised models of abstraction (Block model, Sfard's model and Concrete, representational, abstract framework for example [1, 8, 17, 18];
- tools and problems that are suitable for developing and assessing abstraction skills [5];
- instructional design and practices (e.g. creation activities, comprehension, debugging and testing, pair-programming).

The broad questions we are exploring are:

(1) What do we mean by abstraction skills and how can we determine when undergraduate CS students have acquired them?
(2) How can we adapt suitable abstraction frameworks for the most effective transfer of knowledge and abstraction skills?
(3) Can we use these abstraction frameworks to design a pedagogical framework as described in *Our pedagogical framework* that achieves this aim?

There are three main perspectives that are important here: theoretical computer scientists, computer science education researchers and educators whose modules aim is to develop abstraction skills. Given the diversity of the members, we are able to collate a range of relevant materials in context e.g. exercises, learning materials and instructional materials that intend. We are aligning the materials with the chosen abstraction model(s) and developping pedagogical guidelines to facilitate their application. Following this, we will provide examples of how these can be implemented in teaching and learning environments, offering practical tools for educators.

REFERENCES

[1] Sfard A. 1991. On the Dual Nature of Mathematical Conceptions: Reflections on Processes and Objects as Different Sides of the Same Coin.
[2] Mordechai Ben-Ari. 1998. Constructivism in Computer Science Education. *SIGCSE Bull.* 30, 1 (mar 1998), 257–261. https://doi.org/10.1145/274790.274308
[3] Mirolo C., Izu C., Lonati V., and Scapin E. 2021. Abstraction in Computer Science Education: An Overview.
[4] J Crossley. 2023. How Do Students Conceptualize and Represent Abstract Ideas? An Initial Exploration.. In *International Conference on Computing Education '23'* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 82–86. https://doi.org/10.1145/3568812.3603455
[5] Quintin Cutts, Sarah Esper, Marlena Fecho, Stephen R. Foster, and Beth Simon. 2012. The abstraction transition taxonomy: developing desired learning outcomes through the lens of situated cognition. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (Auckland, New Zealand) *(ICER '12)*. Association for Computing Machinery, New York, NY, USA, 63–70. https://doi.org/10.1145/2361276.2361290
[6] Ginat D. 2022. Abstraction, Declarative Observations and Algorithmic Problem Solving. *Informatics in Education* 20 (2022), 567–582. Issue 4.
[7] Ginat D. and Blau Y. 2017. Multiple Levels of Abstraction in Algorithmic Problem Solving.. In *Proceedings of Special Interest Group in Computer Science Education Technical Symposium*. Association for Computing Machinery.
[8] Fyfe E. and Mitchell N. 2018. Making "concreteness fading" more concrete as a theory of instruction for promoting transfer. *Educational Review* (2018).
[9] Orit Hazzan and Jeff Kramer. 2016. Assessing abstraction skills. *Commun. ACM* 59, 12 (dec 2016), 43–45. https://doi.org/10.1145/2926712
[10] Crossley J., Begum M., and Wood J. 2023. Qualitative insights into abstraction skills from exam scripts.
[11] Hill J., Houle B., Merritt S., and Stix A. 2008. Applying abstraction to master complexity.. In *In Proceedings of the 2nd international workshop on The role of abstraction in software engineering (ROA '08)*. Association for Computing Machinery., New-York, USA., 15–21.
[12] Piaget J. 1972. *The Principles of Genetic Epistemology*. Routledge and Kegan Paul.
[13] Perrenet J. and Kaasenbrood E. 2006. Levels of abstraction in students' understanding of the concept of algorithm: the qualitative perspective. *SIGCSE Bulletin* 38 (2006), 270–274. Issue 3.
[14] Wing J. 2006. Computational Thinking. *Commun. ACM* 49 (2006), 33–35. Issue 3.
[15] Begum M., P. Haglund, Korhonen A., Lonati V., Monga M., Strömbäck F., and Tilanterä A. 2023. Empirical Evaluation of a Differentiated Assessment of Data Structures: The Role of Prerequisite Skills. *Informatics in Education* (2023).
[16] Hazzan O. 2002. Reducing abstraction level when learning computability theory concepts.. In *Proceedings of Conference on Innovation and Technology in Computer Science Education*.
[17] Lister R., Clear T., Simon, Bouvier D., Carter P., Eckerdal A., Jacková J., Lopez M., McCartney R., Robbins P., Seppälä O., and Thompson E. 2009. Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer. *ACM Special Interest Group in Computer Science Education Bulletin* 41 (2009), 156–173. Issue 4.
[18] Carsten Schulte. 2008. Block Model: an educational model of program comprehension as a tool for a scholarly approach to teaching. , 12 pages. https://doi.org/10.1145/1404520.1404535
[19] David Statter and Michal Armoni. 2020. Teaching Abstraction in Computer Science to 7th Grade Students. *ACM Trans. Comput. Educ.* 20, 1, Article 8 (jan 2020), 37 pages. https://doi.org/10.1145/3372143
[20] David Tall and Michael Thomas. 2022. Intelligence, Learning and Understanding in Mathematics A Tribute to Richard Skemp.
[21] Jane Waite. 2017. Pedagogy in teaching Computer Science in schools: A Literature Review.