Poster: a Conceptual Model for Cooperative Thinking

Daniel Russo
University of Bologna - DISI
daniel.russo@unibo.it

Marcello Missiroli
University of Bologna - DISI
marcello.missiroli@unibo.it

Paolo Ciancarini
University of Bologna - DISI
paolo.ciancarini@unibo.it

ABSTRACT

Training computer scientists to address wicked problems means to focus respectively on the individual capability to think in a computational-oriented way (i.e., Computational Thinking), and on the social dimension of coding (i.e., Agile Values). In this study we propose the conceptual model of *Cooperative Thinking*, a new education construct of team-based computational problem solving. Cooperative Thinking is not only the sum of Computational Thinking and Agile Values, rather it is a new overarching competence suitable to deal with complex software engineering problems. We suggest to tackle the Cooperative Thinking construct as an education goal, to train new generations of software developers to Pareto-optimize both their individual and teaming performances.

CCS CONCEPTS

• **Applied computing** → **Education**; • **Software and its engineering** → *Software creation and management*; • **Human-centered computing** → Collaborative and social computing;

KEYWORDS

Computational Thinking, Agile Values, Cooperative Thinking.

ACM Reference format:
Daniel Russo, Marcello Missiroli, and Paolo Ciancarini. 2018. Poster: a Conceptual Model for Cooperative Thinking . In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
https://doi.org/10.1145/3183440.3195062

1 CONTEXT

New skills are required for the future workforce to get employed in the software engineering domain [6]. Future workers will need to think differently, to solve their working tasks. Indeed, tasks solved by software systems are becoming more complex by the day. Some problems in the real world can be classified as *wicked problems* which usually do not have an unique solution but many Pareto-optimal ones.

Accordingly, the education system needs to train students on such new skills. Novel initiatives were promoted by institutions in several countries, like for instance the initiatives *"21st century skills"* [10] in the US and *"Europe's Key skills for Lifelong Learning"* [5] that prompted the redefinition of Computer Science and Software Engineering curricula:

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195062

Computing proficiency became a traversal skill for all domains, complementing the soft skill areas[1]. Modern education theories, such as Constructionism , promote critical thinking as opposed to mere memorization; teaching practices such as Cooperative Learning [7] and Problem-based learning also introduce organizational and social skills in the educational process. However, a grounded analysis about the educational constructs used in Computational Thinking (CT) and Agile Values (AV) is missing in literature. Single constructs were presented, like Computational Thinking [12] for the Computer Science domain, in general, and Agile Values [4] for the Software Engineering one in particular. Surprisingly, a study about a comprehensive educational implementation of such constructs is lacking. With regard to software engineering education, CT and AV represent core skills of software development: the individual ability to produce computationally efficient code and the social ability to interact with peers and stakeholders to deliver effective software.

We argue that these two core skills are part of the higher level skill of *Cooperative Thinking* (CooT), which is, in our view, *the ability to describe, recognize, decompose problems and computationally solve them in teams in a socially sustainable way* [9].

To be a valuable construct, CooT should not be just the sum of two constructs, rather it should "explain" other crucial educational constructs. Consequently, we propose a conceptual model, explaining the theoretical motivations of our hypotheses.

2 RESEARCH MODEL AND HYPOTHESES

Every theoretical construct, as the relations among constructs, needs to be grounded in literature, since the epistemological view of science is not disruptive but incremental. Accordingly, we are now presenting the conceptual model of Cooperative Thinking, based on our scholarly experience and related works.

Effect of Computational Thinking on Cooperative Thinking. In order to enhance the new construct CooT, individual CT skills need to be developed to interact in a constructive way within the group, to suggest useful insights. After Wing's paper [12], several frameworks have been proposed to operationalize it in the daily educational system [2]. The general idea is to educate students to think in a computational–friendly way to improve coding and problem–solving tasks. As such, it is a pivotal individual skill–set that any future worker will bring to its team. Team performance is strictly related to quality of its individual assets [3]. Therefore, the quality of the developed CT skills will affect positively future teams performance.

According to this background, we formulate our first hypothesis:

H_1: Computational Thinking positively influences Cooperative Thinking

[1] https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Effect of Agile Values on Cooperative Thinking. While Computational Thinking is the individual skill useful to solve problems, Agile Values educate people to team up. Agile Values offer a variety of points of view required to solve difficult or wicked problems. To such problems there is no single "best solution", but several Pareto-optimal ones which may change over time.

With particular regard to software engineering, complex system design is a typical wicked problem [13]. Satisfying unpredictable customer's expectations and requirements becomes harder and harder at a steady path, which is beyond the limit of solvability for any single programmer. Delivering *valuable* software *on time* has been one of the major efforts of software development methodologies in the last years. Although the definition of "on time" may look clear (since it is related to a deadline), it is strictly correlated to "valuable", which is a more vague definition. With reference to the ISO 25010:2011 standard on software quality, the customer may perceive as valuable aspects related to the Quality in Use dimension. Nevertheless, a software with high Quality in Use but a low e.g., maintainability (which is related to the Product Quality) could not be really defined "valuable". The aspect of maintainability may be related to poor refactoring due to time constraints.

In this example, it is clear that value and time are two sides of one coin. Mastering such challenges requires a specific skill–set. The Agile Manifesto proposed a new perspective on software development, based on values that clashed with the established culture of time, based on linear hierarchies, top-down decision making and, in general, accepting the current system without voicing dissent or criticism [1]. The most significant change is the paramount importance assigned to *communication and social interaction*, superseding the internal organizational rigidity, documentation, contracts, roles, and more. In time, this led to the formalization important concepts (such as changing requirements, self-organizing teams, personal responsibility, ...) and programming practices (pair programming, test-first development, continuous integration, ...). Agile has proven in several contexts its usefulness, and it is now an established development model and its adoption is steadily growing [8].

Accordingly, Agile Values are an important skill–set for Cooperative Thinking, leading to our second hypothesis:

H_2: Agile Values positively influence Cooperative Thinking

Effect of Cooperative Thinking on Complex Problem Solving. As proposed with H_1 & H_2, the construct Cooperative Thinking is mainly explainable with Computation Thinking and Agile Values. According to our empirical experience [11], it is not just the sum of these constructs. Rather it is a useful proxy to develop further fundamental skills. Complex Problem Solving is the most suited construct to identify the most relevant skills, as suggested by [6]. According to its definition it is "Developed capacities used to solve novel, ill-defined problems in complex, real-world settings". In other words, it is another way to define wicked problems.

Crucial future skills can not be taught with an old–fashioned curriculum. From a pedagogical perspective we started questioning ourselves how to teach best students to the management of wicked problems. With regard to Computational Thinking and Agile Values we realized that alone, they are not sufficient. CT deals with individual capabilities and is deeply routed in the traditional educational system of "solo" learners. On the other hand, AV *per se*, are not enough deal with such problems. Good interaction is a valuable driver but not the asset to solve wicked issues.

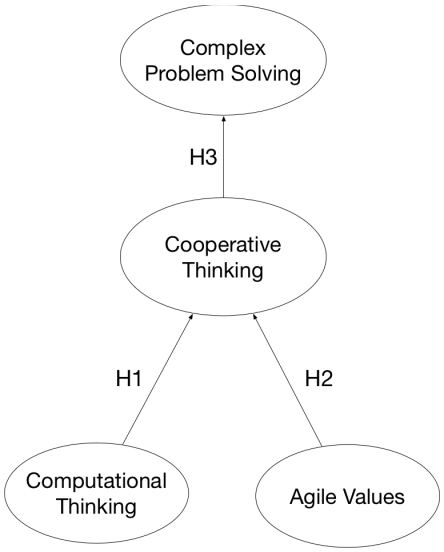

Figure 1: Theoretical framework and hypotheses

The idea of Cooperative Thinking, is that of a construct which is able to teach students to tackle Complex Problem Solving as a proxy of wicked problems. Therefore, our last hypothesis is:

H_3: Cooperative Thinking positively influences Complex Problem Solving

Future work. This conceptual model is preliminary for an empirical validation e.g., with Structural Equation Modeling.

REFERENCES

[1] Agile Alliance. 2001. Agile manifesto. *Online at http://www. agilemanifesto. org* 6, 1 (2001).
[2] ACM Computer Science Teachers Association. 2011. Operational Definition of Computational Thinking. (2011). http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf
[3] M. R Barrick, G. L Stewart, M. J Neubert, and M. K. Mount. 1998. Relating member ability and personality to work-team processes and team effectiveness. *Journal of Applied Psychology* 83, 3 (1998), 377–391.
[4] K. Beck and C. Andres. 2004. *Extreme programming explained: embrace change.* Addison–Wesley.
[5] European Communities. 2007. Key competences for lifelong learning: European Reference Framework. (2007).
[6] World Economic Forum. 2016. *The Future of Jobs: Employment, Skills and Workforce Strategy for the Fourth Industrial Revolution.*
[7] D. Johnson et al. 1994. *Cooperative learning in the classroom.* ERIC.
[8] Y. Lindsjørn, D. IK Sjøberg, T. Dingsøyr, G. R Bergersen, and T. Dybå. 2016. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software* 122 (2016), 274–286.
[9] M. Missiroli, D. Russo, and P. Ciancarini. 2017. Cooperative Thinking, or: Computational Thinking meets Agile. In *Proc. CSEE&T*. IEEE.
[10] The Glossary of Education Reform. 2016. 21st century skills. http://edglossary.org/21st-century-skills/. (2016).
[11] D. Russo, M. Missiroli, and P. Ciancarini. 2018. *Cooperative Thinking: Analyzing a New Framework for Computer Science Education.* Technical Report.
[12] J. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.
[13] R. T Yeh. 1991. System development as a wicked problem. *International Journal of Software Engineering and Knowledge Engineering* 1, 02 (1991), 117–130.