The Journal of Systems and Software 157 (2019) 110401

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Cooperative Thinking: Analyzing a new framework for software engineering education

Paolo Ciancarini[a,b,*], Marcello Missiroli[b], Daniel Russo[c]

[a] Innopolis University, Russia
[b] University of Bologna, Italy
[c] Lero - The Irish Software Research Center & School of Computer Science and Information Technology, University College Cork, Ireland

ARTICLE INFO

Article history:
Received 14 November 2018
Revised 23 May 2019
Accepted 21 August 2019
Available online 22 August 2019

Keywords:
Partial Least Squares
Structural Equation Modelling
Multivariate Analysis
Latent Variable Analysis
Empirical Software Engineering
Computer Science Education
K-12
High School
Software Engineering Education
Computational Thinking
Agile
Cooperative Thinking

ABSTRACT

Computational Thinking (CT) and Agile Values (AV) focus respectively on the individual capability to think algorithmically, and on the principles of collaborative software development. Although these two dimensions of software engineering education complement each other, very few studies explored their interaction. In this paper we use an exploratory Structural Equation Modeling technique to introduce and analyze *Cooperative Thinking* (CooT), a model of team-based computational problem solving. We ground our model on the existing literature and validate it through Partial Least Square modeling. Cooperative Thinking is new competence which aim is to support cooperative problem solving of technical contents suitable to deal with complex software engineering problems. This article suggests to tackle the CooT construct as an education goal, to train students of software development to improve both their individual and teaming performances.

1. Introduction

According to the World Economic Forum, current technological trends - like mobile Internet and cloud technology, advances in Big Data, advanced robotics and autonomous transport, artificial intelligence and machine learning, advanced manufacturing and 3D printing and High Performance Computing, new materials, biotechnology and genomics, just to cite a few, propose novel problems to both users and developers (WEF, 2016). According to this vision, workers will need to think differently, to solve their working problems in a context where software systems are becoming more complex day by day. Some problems in the real world can be classified as *wicked problems* which can be considered as complex real-world problems (Rittel and Webber, 1973). In other words, these problems outline trade-off situations, where a selection of alternatives is needed: each option is first assessed, and then a subset of those options is identified, with the property that no other option can outperform any of the chosen options.

Accordingly, the education system needs to train students on such new challenges. Novel initiatives were promoted by institutions in several countries, like for instance the US *"21st century skills"* (Vv.Aa., 2016a) and *"Europe's Key skills for Lifelong Learning"* (EC, 2017) initiatives, that prompted the redefinition of computer science curricula:

> *[...] to empower all [...] students to learn Computer Science and be equipped with the computational thinking skills they need to be creators in the digital economy, not just consumers, and to be active citizens in our technology-driven world. Our economy is rapidly shifting, and both educators and business leaders are increasingly recognizing that Computer Science is a "new basic" skill necessary for economic opportunity and social mobility.*[1]

The idea of a "new basic skill", according to this view, derives from the fact that computational proficiency became a traversal

* Corresponding author at: DISI - Mura Anteo Zamboni, University of Bologna, 7. 40126 Bologna - Italy.
E-mail addresses: paolo.ciancarini@unibo.it (P. Ciancarini), marcello.missiroli@unibo.it (M. Missiroli), daniel.russo@lero.ie (D. Russo).

https://doi.org/10.1016/j.jss.2019.110401
0164-1212/

[1] https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all. Accessed on 22.01.2018.

skill for all domains, complementing the soft skill areas. Modern education theories, such as Constructionism (Papert and Harel, 1991), promote critical thinking as opposed to mere memorization; teaching practices such as Cooperative Learning (Johnson et al., 1994) and Problem-Based Learning (Hung et al., 2008) also introduce organizational and social skills in the educational process. These theories are becoming widely spread, although some scholars have mixed-feelings about them, suggesting that e.g., memorization is a prerequisite for critical thinking (Kirschner et al., 2006; Sweller, 2004).

Single constructs were presented, like Computational Thinking (Wing, 2006) for the computer science domain, in general, and Agile Values (Beck and Andres, 2004) for the software engineering one in particular. With regard to the pedagogical perspective, also relevant efforts to support collaboration in Computational Thinking, like the 4C paradigm (Critical thinking and problem solving, Communication, Collaboration, and Creativity and innovation) (Trilling and Fadel, 2012) have been made. Similarly, tools like Scratch (Maloney et al., 2010), which "primary goal is not to prepare people for careers as professional programmers [...]"(Resnick et al., 2009, p. 60), have been developed to trigger relevant constructs, like Computational Thinking, in schools.

Computational Thinking (CT) and Agile Values (AV) represent complementary skills of computer science education for software development (Missiroli et al., 2017b): respectively, the individual ability to produce computationally efficient code, and the social ability to interact with both peers and stakeholders to deliver valuable software. Nevertheless, CT and AV have also practical implications in the broader computer science domain. The rise of complexity (and wicked problems) is not only a problem of software engineering, but engages all computer science areas. The interdisciplinary interaction between different hardware and software components, along with a context-dependent knowledge is a common scenario for most areas. As an illustrative example, IoT is moving to new paradigms due to raising complexity of computing (e.g., fog computing, context-aware computing) (Montori et al., 2018). Here, the role of teams is crucial to address these topics, since they are both interdisciplinary and complex.

We argue that these two core skills are part of the higher level competence 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* (Missiroli et al., 2017b). Computational Thinking is the skill of understanding problems and applying, assessing, and producing a solution in form of an algorithm. It is a basic skill that comes to the meaning of problem solving and points out that it is necessary to understand what the problem is before developing a solution while solving a problem according to a specific point of view (Korkmaz et al., 2017). However, an individual which masters the Computational Thinking skill is not necessarily able to cooperate with other individuals to solve complex problems (Roman-Gonzalez et al., 2018). Accordingly, we worked on the concept of Cooperative Thinking working with teams of students in high schools and in university courses. Our initial idea was to exploit an agile approach to let the teams solve problems requiring Computational Thinking (Missiroli et al., 2016a). We started with teams composed of pairs, then scaled to self-organizing groups of up to six students. We realized that, when working in team on complex problem solving, *social sustainability* is important: in particular we found that heterogeneous groups are more effective than homogeneous groups (Missiroli et al., 2016a). We noticed that such groups were able to handle complex problems more effectively, due to their ability to team up through peer education and communication. Especially for software developers, communication structures are essential to understand the way they design software: this is called the Conway law (Conway, 1968). Since communication impacts the way they design software systems, it is necessary to educate developers to properly manage their social organization of work (i.e., dealing with customers, rely upon fellow developers, be able to discuss algorithms, etc.). It should be socially sustainable, since a developer should be able not only to deliver her specific task (e.g., developing some piece of code), she should also interact effectively with her social context (e.g., internal and external project's stakeholders, laws and regulations). Educating students to deal responsibly with their social context means to make them aware that a socially sustainable work organization is important to solve complex problems. We are less interested in educating solo developers who provide fast algorithmic solutions, regardless of their social communication structures. Indeed, social sustainability is a new element which is complementary to both Computational Thinking and Agile Values.

It focuses on cooperative problem solving of technical contents. So, this competence is derived from both CT and AV and is able to address complex problem solving skills. However, CooT is not just the sum of two constructs, but it is an autonomous educational construct which builds on Complex Negotiation, Continuous Learning, Group Awareness, Group Organization, and Social Adaptability, as will be explained in Section 6.1. To evaluate this assumption, we validated CooT through Partial Least Squares Structural Equation Modeling (PLS-SEM), a method that has also been used to analyze the relation between Computational Thinking skills and different work and school specific variables such as IT usage experience or IT academic success (Durak and Saritepeci, 2018). This research method enables researchers to assess if the relationships among different theoretical constructs are statistically significant in the surveyed population.

This paper is organized as follows. In Section 2 we present the related literature. Subsequently, in Section 3 we discuss our research model with the underlying hypothesis. Then, we describe our research methodology in Section 4, along with a brief explanation of PLS. Afterwards, we validate the results obtained with PLS in Section 5. The analysis of our findings with the study limitations is in Section 6. Finally, we outline future works and our conclusions in Section 7.

2. Background and related work

There is a growing belief that complex problem solving, critical thinking, creativity, people management, and coordinating with others will become the most important job skills by 2020 (WEF, 2016). According to the World Economic Forum (WEF), future companies will actively search for employees who can master "capacities used to solve novel, ill-defined problems in complex, real-world settings" and "motivate, develop and direct people as they work, identifying the best people for the job, also adjusting actions in relation to others' actions" (WEF, 2016). So, skills to think in a computational friendly way and to solve them in a social and sustainable manner are both required. Apparently, CT and AV skills are strictly connected for companies, as suggested by WEF (2016).

Since 1945, several scholars have been theorizing *ante litteram* about Computational Thinking, most notably by Papert (1980), and Polya (1957). The idea of "algorithm" became popular after 1960 when Katz suggested that automated processes would spread well beyond the computer science domain and would influence all fields (Katz, 1960).

In 2006, Jeannette Wing's paper popularized the concept of Computational Thinking (Wing, 2006), portrayed as a fundamental skill in *all* fields, not only in computer science. It is a way to approach complex problems, breaking them down in smaller problems (decomposition), taking into account how similar problems have been solved (pattern recognition), ignoring irrelevant

information (abstraction), and producing a general, deterministic solution (algorithm). Today, governments are realizing its importance, and update school programs worldwide (like the US initiative *"21st century skills"*, Vv.Aa., 2016a).

However, more and more scholars argue whether the CT concept is too vague to have a real effect (Denning, 2017). Denning claims that CT is too vaguely defined and, most important in an educational context, its *evaluation* is very difficult to have practical effects (Denning, 2017). This same idea can be found in the CS Teaching community. Barr and Stephenson (2011) and Hoskey and Zhang (2017), for example, try to decompose the CT idea itself, in order to have an operative definition. Henderson (2009) notes that computing education has been too slow moving from the computing programming model to a more general one. Blackwell et al. (2008) even wonders if the CT concept is at all useful in computer science, since it puts too much importance on abstract ideas. We also noted that a part from some works (Howard et al., 1996; Thomas et al., 2002; Allert, 2004), there is not much research on CT and learning styles.

Though Agile development is eventually going mainstream in the professional world, *teaching* the Agile methodology is still relatively uncommon, especially at the K-12 level; there are a few exceptions (Steghöfer et al., 2016; Kropp and Meier, 2014). Indeed, university curricula typically focus on Waterfall-like development models (Kropp and Meier, 2013). Often, especially experienced practitioners, learn Agile "in the field", or after attending *ad hoc* seminars, since they did not have the chance to learn it while in education. Interest in Agile is however rising, and curricula are being updated to reflect such trend (Steghöfer et al., 2016; Kropp and Meier, 2014). A comprehensive proposal has been advanced by Meerbaum-Salant and Hazzan (2010), where the "Agile Constructionist Mentoring Methodology" and its year-long implementation in high school is presented. It considers all aspects of software development, with a strong pedagogical support.

In the last years we gathered several insights along our research journey on computer science education (Missiroli et al., 2016a; 2016b; 2017a). From our experience, we realized that computer science skills, like programming, are typically taught at an individual level. There are several reasons why this is the case. Probably, the main reason is students' assessment. Since it is much harder to trace the acquired knowledge of each single student while working in a group, the most straightforward option is to consider the class as a set of individuals. Indeed, there are also other reasons, like the necessity to tailor Individual Learning Plans, especially for students with special needs (Carroll, 1990), although recent research showed that heterogeneous groups outperform homogeneous ones (also those composed by very good students) (Missiroli et al., 2016a).

Surprisingly, educational approaches to convey computer science students with a broader set of skills (both of social and technical nature) are not so common in our community with few exceptions, like (Carter, 2011; Burden et al., 2019). Notably, Burden et al. (2019) provided insightful evidence, suggesting that Agile methods in project-based classes are an opportunity to experience entrepreneurial skills during software engineering classes. The authors suggest that using Agile approaches in project-based courses stimulates opportunities for entrepreneurial experiences in software engineering courses since they can be implemented in student projects to lead ideas into action.

Generally speaking, the traditional educational paradigm is not well tailored to educate people to handle complex issues or *wicked problems* (Buchanan, 1992). PISA-like evaluations are meaningless to determine the educational system's efficiency in this respect, since they consider the individual performance of students. So, the gap between students' formal educational background and real life wicked problems and the related complex task becomes larger as the level of predictability decreases and uncertainty increases (Raskino and Waller, 2016).

Some studies tackled the idea that hard skills expertise should be complemented with soft skills, possibly introducing active and cooperative learning to CS (Johnson et al., 1994). For example, in Rivera-Ibarra et al. (2010), a long list of so-called soft skills expertise is paired with various developer's roles. In Carter (2011), the problem is well analyzed, but arguably the proposed solution is not comprehensive. Meier et al. (2016) presents an example of how to promote cooperation within a software project; however generalizing the proposed scheme seems difficult. Notably, team-based learning (Michaelsen et al., 2002) has been applied to computer science courses (Lasserre and Szostak, 2011), as also Project-Based Learning (Steghöfer et al., 2018) although they are mainly concentrated in Scandinavian countries.

The scholarly debate did substantial contributions to our understanding of Computational Thinking and Agile Values. However, emergent educational practices, like Cooperative Thinking, Missiroli et al. (2017b) requires a more in-depth analysis. Therefore, we have recently designed a research model to investigate Cooperative Thinking (Russo et al., 2018). In Russo et al., we defined a conceptual model for Cooperative Thinking, providing a theoretical support to the construct hypothesized in Missiroli et al. (2017b). In concrete terms, we discussed the relevant theoretical hypotheses related to the Cooperative Thinking construct. Therefore, we used a multivariate analysis technique to test our hyphoteses, to provide the community with a first theory of the observed phenomenon, i.e., also known as "soft theory" (Russo and Stol, 2019).

This stream of research is based on several experiments (Missiroli et al., 2016a; 2016b; 2017a). suggesting that effective coding teamwork in educational environments leads to improved learning outcomes and even to software of better quality. Nevertheless, good teamwork is not sufficient, *per se*, to solve complex tasks – individual problem solving competencies are also needed. In previous works, we found that the best outcomes were provided in cases where both such competences (i.e., teamwork and problem solving skills) were effectively implemented (Missiroli et al., 2016a). Recently, these problems have been addressed by theoretical contributions (Missiroli et al., 2017b; Russo et al., 2018), but they have never been validated. Still, substantial questions remain open in literature, like what is the best way to educate CS students to manage both teaming and software development skills, or the best educational practices to use in this regard. Therefore, our aim is to address this gap.

3. Research model and hypotheses

Based on the prior discussion, we forward our basic thesis. Future workers will need a new set of skills to be competitive on tomorrow's job market. *Ad hoc* educational curricula need to be developed to prevent skill shortage. Apparently, CT and AV alone are not sufficient to educate students to solve wicked problems (Weber and Khademian, 2008). The development of a new overarching competence may lead students to describe, recognize, decompose problems and computationally solve them in teams in a socially sustainable way. This competence, which we named Cooperative Thinking, is not just the sum of the two underlying constructs of CT and AV. We propose to consider it as a social dimension of computer science education.

For the sake of this paper, we used the definition of Complex Problem Solving to identify the most relevant skills, as suggested by the WEF (2016).

This is an exploratory study to assess if the formalized constructs have a significant relationship with each other. From an operational perspective, constructs are phenomena which can only

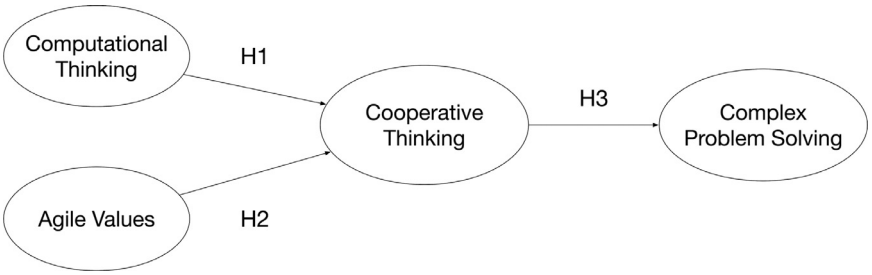


Fig. 1. Theoretical framework and hypotheses.

be measured through latent variables (like project success, complexity, commitment, or values, Batra, 2018), which are not directly observable but inferred from other directly observed variables. As a Structural Equation Model based study, constructs are grounded in literature or experience (Hair et al., 2016). Therefore, we are hypothesizing relationships which have a theoretical explanation but were never assessed, which is an important novel contribution of this paper.

In the next subsections we are motivating our hypotheses, supported by Russo et al. (2018).

3.1. Effect of Computational Thinking on Cooperative Thinking

As explained in Section 2, in order to enhance the new construct Cooperative Thinking, some individual Computational Thinking skills need to be developed to interact in a constructive way within the group, to suggest useful insights. Following Wing (2006), several frameworks have been proposed to operationalize Computational Thinking in an educational system (Vv.Aa., 2016b; 2015; 2011). The general idea is to train students to think in a computational-friendly way to improve their problem-solving skills. 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 members (Barrick et al., 1998). Therefore, the quality of the developed CT skills will affect positively the performance of the team in the future.

According to this background, we formulate our first hypothesis:

H_1 : *Computational Thinking positively influences Cooperative Thinking*

3.2. Effect of Agile Values on Cooperative Thinking

While Computational Thinking is the specific skill useful to individuals to solve problems, Agile Values educate people to work together. Agile Values offer a variety of points of view useful to solve difficult or wicked problems. Usually there is no single "best solution" to such problems, but several ones, whose value moreover may change over time—as is the case in the field of Science and Business (Camillus, 2008).

With particular regard to software engineering, the design of a complex system whose requirements are unstable is a typical wicked problem (Yeh, 1991). Satisfying unpredictable customer's expectations and ephemeral requirements 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 (Dingsøyr and Lassenius, 2016). 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 (trivial) 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 multi-level hierarchies, top-down decision making and, in general, accepting the given methods without voicing dissent or criticism (Alliance, 2001). The most significant change invoked by the Agile movement is the paramount relevance assigned to *communication and social interaction*, superseding any internal organizational rigidity, documentation, contracts, roles, and more.

This led to the formalization of important concepts (such as changing requirements, self-organizing teams, personal responsibility, ...) and programming practices (pair programming, test-first development, continuous integration, ...). The Agile approach has proven in several contexts its usefulness, and it is now an established development model and its adoption is steadily growing (Lindsjørn et al., 2016).

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

H_2: *Agile Values positively influence Cooperative Thinking*

3.3. Effect of Cooperative Thinking on Complex Problem Solving

As proposed with H_1 and H_2, the construct Cooperative Thinking is mainly explainable with Computation Thinking and Agile Values. Nevertheless, we do not believe that it is just the sum of these constructs. Rather it is a useful proxy to develop further fundamental skills.

The intuition is that some crucial future skills cannot be taught with an old-fashioned curriculum. The most significant skill for future workers in 2020 is, according to the World Economic Forum, *Complex Problem Solving* (WEF, 2016). 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 solve wicked problems.

From a pedagogical perspective we started questioning ourselves how to train our best students to manage wicked problems. With regard to Computational Thinking and Agile Values we realized that, separately, 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 to deal with such problems. Good social interaction is a valuable driver but not the asset to solve wicked issues.

The idea of Cooperative Thinking, as defined in Section 2, 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*

The relationships among our three hypotheses can be represented as in Fig. 1.

4. Research design

Structural Equation Modeling (SEM) is strongly influenced by Popper's post-positivist view, according to which social observations should be treated as entities like physical phenomena (Popper, 2005). Using SEM the researcher is detached from the observed constructs, as social science inquiry should be objective and hypotheses should be empirically validated to justify them. Typically research outcomes obtained with SEM are generalizable, independently from time and context (Nagel, 1986).

As this is an exploratory study we are here interested to test the significance of the proposed model. For this reason, post-positivism is the best suited meta-theoretical stance, since we are dealing with the falsification (i.e., significance verification) of our hypotheses. As researchers, we obviously have our epistemological bias, which usually remain hidden or implicit, even if they deeply influence our research (Slife and Williams, 1995). Therefore, empirical (i.e., statistical) procedures are of greatest importance to mitigate researcher's biases (Popper, 2005).

4.1. Research questions

We are interested in testing these two assumptions: (a) is CooT grounded in empirical evidence, and (b) does it address key constructs, like Complex Problem Solving? This leads us to our first research question:

> RQ_1: Is *Cooperative Thinking* grounded as a new overarching theoretical construct in Computational Thinking and Agile Values?

Our second research question regards the "explanatory" power of our construct:

> RQ_2: Is *Cooperative Thinking* a significant construct to teach students how to deal with wicked problems?

During this research journey, some explicit dimensions which were initially implicit emerged. Thus, beyond the proposal of a new construct which should be considered for curricular purposes, we validate it through a well established statistical method.

4.2. Partial Least Square path modeling

The use of Partial Least Squares Structural Equation Modeling (PLS-SEM) for the validation of latent unobserved variables with multiple observed indicators (Chin, 1998b) is an emerging research trend within the computer science education domain (Lu et al., 2007; Seman et al., 2018; Goggins and Xing, 2016; Shakroum et al., 2018; Fraj-Andrés et al., 2018). It has been developed by Wold for the analysis of high dimensional data (i.e., with a high number of independent variables) in low structured environments, typical of social science settings (Wold, 1974; 1983). SEM techniques in general, and PLS in particular, is able to answer a set of interrelated research questions in one comprehensive analysis (Gefen et al., 2000). Other research communities have even a longer tradition with PLS-SEM and made several advances for theoretical model testing, in Management (Hulland, 1999), Information Systems Research (Dibbern et al., 2004), and Organizational Behavior (Higgins et al., 1992). "SEM has become *de rigueur* in validating instruments and testing linkages between constructs" (Gefen et al., 2000, p. 6), since it allows to distinguishes between measurement and structural models, taking also measurement error into account.

Practically speaking, any structural equation model is composed by two sub-models: a structural model and a measurement model. The structural model designs the relationships between the different constructs; while the measurement model provides the measures for the different latent variables. In order to have a reliable estimation of the hypothesized relations among the latent constructs (in the structural model), the measures which define the different constructs has to be grounded on auxiliary theory (in the measurement model), since "without this auxiliary theory, the mapping of theoretic constructs onto empirical phenomena is ambiguous, and theories cannot be empirically tested" (Edwards and Bagozzi, 2000, p.115).

SEM distinguish itself between two families: the first one are covariance-based techniques (CB-SEM); the second one are variance-based techniques, among which Partial Least Squares (PLS) path modeling is the most used one (Henseler et al., 2009). CB-SEM is considered a more conservative approach, designed for confirmatory and theory-testing research; while PLS-SEM aim is develop new theory or predictive applications (Henseler et al., 2009). This is because CB-SEM has more stringent assumptions (e.g., normal distribution and high sample sizes), since it minimizes Type I and Type II errors. This is not the case of the PLS-SEM algorithm, which has exploratory purposes.

Operational research scholars consider PLS-SEM as a "silver bullet" for estimating predictive models in many theoretical models and empirical data situations (Hair et al., 2011). Indeed, it is flexible in the construction of unobserved latent variables and modeling relations among different predictor criteria and variables (Chin and R. Newsted, 1996).

4.3. Scale development

As any SEM study, the scale was developed with the greatest care with the help of auxiliary theory (Edwards and Bagozzi, 2000). Following the example of Lu et al. (2007), latent variables (i.e., constructs) were measured through uni-dimensional items (in form of statements), which selected informants answered according to the statement's level of agreement on a 7-point Likert scale. Constructs and items are represented in Table 1.

All constructs in the model are "reflective". Indeed, latent variables can be measured in either reflective or formative ways (Diamantopoulos and Siguaw, 2006; Gudergan et al., 2008). We use *reflective* ones when items are caused by the latent variable (i.e., their covariance), or in other words, when they represent the effects of the underlying construct so that causality is from the construct to its items. They can be considered as a representative sample of all the possible items available within the conceptual domain of the construct (Hair et al., 2016).

We wanted to ground the definition of each construct to frameworks established in literature. As no universally accepted framework exist for any of them, we had to pick the framework best suited to our needs. In particular, for *Computational Thinking* we used the framework proposed by Computing at School, a subdivision of the British Computer Society (Csizmadia et al., 2015),

Table 1
Items list.

Construct [source]	Labels	Items	Questions
Cooperative Thinking (Missiroli et al., 2017b)	COOT_1	Complex negotiation	During design, I like to discuss with people who have different ideas, in order to develop the best solution.
	COOT_2	Continuous learning	Programming in team taught me something I didn't know.
	COOT_3	Group awareness	I like to be part of a software developing team.
	COOT_4	Group organization	When I work in team, results are better than when I work alone.
	COOT_5	Social sensitivity	During development, I work fine even with teammates with whom I have personal difficulties.
Agile Values (Beck and Andres, 2004)	AV_1	Timeboxing and estimation	I can estimate precisely the time needed to complete a developing task.
	AV_2	Simple solution	It is important to find a solution, regardless how, also if not generally applicable.
	AV_3	Programming practices	I use Agile practices during software development
	AV_4	Agile SDLC	I prefer Agile methods to traditional ones.
	AV_5	On-site customer	When I work in a team, I join frequently conversations with teammates or clients/stakeholders.
	AV_6	Face-to-face communication	I prefer direct, face-to-face, communication to emails or messages.
	AV_7	Courage	During a discussion with teammates, I am able to well defend my point of view.
Computational Thinking (Csizmadia et al., 2015)	CT_1	Logical reasoning	I get good results in logical-mathematical tests and exercises.
	CT_2	Algorithmic thinking	I can usually decompose a problem in precise and sequential steps.
	CT_3	Generalization	I discard details not essential to solving design problems.
	CT_4	Evaluation	I like to modify a working solution to improve it, even risking to waste a lot of time.
	CT_5	Patterns	I can easily identify and evaluate recurring patterns or behaviors.
	CT_6	Decomposition	I can always decompose a complex Problem into simpler ones.
Complex Problem Solving (WEF, 2016)	CPS_1	Curiosity	I'm good at working on problems I never tackled before.
	CPS_2	Creativity	I can solve ill-defined problems.
	CPS_3	Tenacity	I like to solve real, complex problems.

enumerating six basic elements of CT (Generalization, Decomposition to name a few). For *Agile Values*, Kent Beck formalized the construct in Beck and Andres (2004), and defined several key factors needed in efficient software development—related to both the personal and social realm (for example, communication). *Complex Problem Solving* has been defined by the World Economic Forum in his pivotal report of future skill needs (WEF, 2016); these skills are not tied to programming but rather to general personal abilities and attitudes. Finally, Cooperative Thinking is based on the result of our studies about the education of Agile student developers to enhance their Computational Thinking capabilities (Missiroli et al., 2016a; 2016b; 2017a; 2017b; Russo et al., 2018), pointing out the importance of social skills and self-organization in software development.

Items related to the constructs were developed independently by the authors and refined iteratively until full consensus was reached. After that, a pre-test with five potential target respondents (i.e., graduate students) was conducted to test the usability of the survey, its rationale, and also the wording. Usability was assessed positively, while minor rationale and wording issues emerged and were consequently fixed.

4.4. Data collection

First, we ran an *a priori* power test (Faul et al., 2009), to define the minimum sample size for a linear multiple regression F-test, which is a good approximation for a PLS analysis. With an effect size of 15%, and a power of 80%, the minimum sample is 77. Then, we used a stratified convenience sampling technique. The samplehad to represent future software developer professionals (e.g., technical High School and computer science students).

To validate the latent variable, grounded in the conceptual model of Russo et al. (2018) and Missiroli et al. (2017b), we used informants which had been already exposed to both Agile practices and Computational Thinking training along their studies. This procedure supports the idea that Cooperative Thinking is derived from the combination of AV and CT. According to that, we see the improvement of the reliability of our endogenous and reflective constructs. Strata were designed accordingly, focusing on undergraduate and graduate students of some European Universities (Bologna, Modena, Limerick, and Chalmers). We also included a significant strata of High School students which were also exposed both to CT and AV during their education. To any subgroup was assigned an ID code for strata definition.

The respondents' rate was 70%, since the survey was directly administered during class by teaching personnel. To avoid random answers, survey's compilation was on a voluntary basis, minimizing response biases and increasing our internal validity. So, we were able to collect only committed students' answers.

Finally, demographics variables relevant to the context and strata were controlled and represented in Table 2. Survey's design is displayed in Table 3 for the sake of reproducibility. Additionally, Table 3 shows the respondents' breakdown both per country and institution. In total we had 116 respondents, well above the minimum requirement. Undergraduate and Graduate students were 72 (62%), while High School students were 44 (38%). We collected several factors, like the programming experience, completed software projects, Agile method experience, and large team participation to identify the sample's skill-set.

5. Results

In this section we describe our results, which consist in the structural equation model described in Fig. 1, computed through our survey data. In order to minimize possible errors or misspecification and assess the significance of our model, we strictly followed the state to the art evaluation protocol proposed by Hair et al. (2016) to make results consistent with our claims. Thus, to estimate the path weighting scheme we used Smart PLS 3.0 (Ringle et al., 2015). Our model converges after 10 iterations. We applied also non-parametric bootstrapping to obtain standard error's estimates (Chin, 1998a; Efron and Tibshirani, 1994). Blindfold-

Table 2
Demographics.

	%	#
Population		
Grad. and Undergrad. students	62%	72
High school students	38%	44
Programming experience		
Less than 1 year	9%	10
2–3 years	46%	53
4–6 years	27%	31
7–10 years	4%	5
11–20 years	9%	10
21–35 years	3%	4
More than 35 years	3%	3
Complete software projects		
1	10%	12
2–4	43%	50
5–10	30%	35
11–20	5%	6
20+	11%	13
Agile methods experience		
Daily	10%	12
Used in some projects	44%	51
Did some experiment	19%	22
I studied it	27%	31
Largest team participated in		
0–2	6%	7
3–5	37%	43
6–8	40%	46
9–12	8%	9
13+	9%	11

Table 3
Educational institutions breakdown.

Educational institution	Country	#
University of Bologna	Italy	47
University of Modena and Reggio Emilia	Italy	21
IIS Fermo Corni (HS)	Italy	44
University of Limerick	Ireland	6
Chalmers University of Technology	Sweden	22

Table 4
Outer loadings.

	AV	CPS	CT	CooT
AV_3	0,830			
AV_4	0,884			
AV_5	0,655			
CooT_1				0,701
CooT_2				0,693
CooT_3				0,865
CooT_4				0,709
CPS_1		0,801		
CPS_2		0,648		
CPS_3		0,891		
CT_1			0,745	
CT_2			0,756	
CT_5			0,756	
CT_6			0,794	

ing was used to calculate Stone-Geisser's Q square value, which represents an evaluation criterion for the cross-validated predictive relevance of the PLS path model (Geisser, 1974; Stone, 1974).

5.1. Measurement model

All item loadings above the cut-off value of 0.65 were considered, as in Table 4, and were significant at $p < 0{,}001$ (with the only exception of CPS with $p < 0{,}05$, since it is the highest construct). Following Hair et al. (2016), items below the cut-off value were rejected (i.e., AV 1, AV 2, AV 6, AV 7, CT 3, CT 4, CooT 5). The good average of items loading and a narrower range of difference for such an exploratory study provide an adequate base for the items in measuring the underlying construct (Hair et al., 2016). Items are not redundant, since the outer variance inflation factor (VIF) ranges between 1165 and 1832, well below the cut-off value of 5 (Hair et al., 2016). Thus, we conclude to have an appropriate item reliability.

Table 5
Fornell–Lacker Criterion.

	AV	CPS	CT	CooT
AV	0,796			
CPS	0,354	0,786		
CT	0,331	0,612	0,763	
CooT	0,570	0,285	0,397	0,745

Table 6
Heterotrait–Monotrait Ratio of Correlations (HTMT).

	AV	CPS	CT	CooT
AV				
CPS	0,444			
CT	0,456	0,803		
CooT	0,764	0,340	0,513	

The construct reliability and validity is composed by the reliability of constructs, composite reliability and average variance extracted (AVE) (Fornell and Larcker, 1981). To assess the construct reliability we used Cronbach's alpha, which measures the homogeneity of items in a construct based on the assumption that each item in the scale contributes equally to the latent construct. The composite reliability depends on the item loadings estimated in the measurement model to compute the measure of internal consistency (Werts et al., 1974). According to Nunnally (1978) both Cronbach's alpha and composite reliability should have at least a value of 0,70 to be acceptable. Rho$_a$ is another reliability measure developed by Dijkstra and Henseler (2015), according to which the most conservative critical value should be above 0,7. For AVE a value above 0,5 is desirable, since it reflects the variance captured by indicators. If this is the case, it means that the variance captured by indicators is greater than the measurement errors.

We assess the discriminant validity to analyze the relationships between latent variables with both Fornell–Lacker Criterion and Heterotrait–Monotrait Ratio of Correlations (HTMT) (Ringle et al., 2012). According to the Fornell–Lacker Criterion the square root of AVE must be greater than the correlation of the construct with all other constructs in the structural model (Fornell and Larcker, 1981). In this way we can see if constructs do not share the same type of items and are so conceptually different from each other. As shown in Table 5, the lowest square root of AVE is 0,745 (CooT-CooT), which is greater than the highest correlation value of 0,612 (CPS-CT). With regard to HTMT, all values are below the most conservative threshold of 0,85 (Henseler et al., 2015), as shown in Table 6.

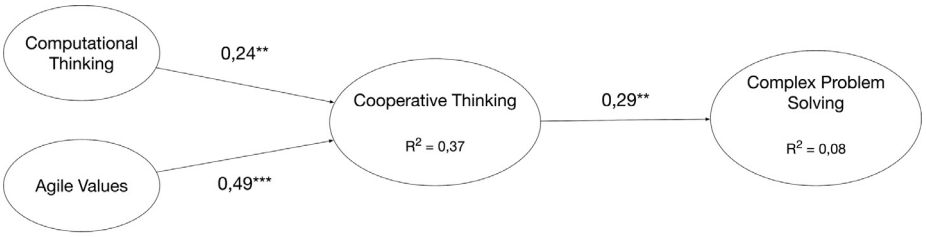
We conclude that the measurement model provides evidence of adequate reliability and validity for the reflective constructs.

5.2. Structural model

We assess the validity and exploratory power of the structural model.

The first step is to test whenever the inner variance inflation factor values (VIF) are below the threshold value of 5 to discard redundant inner–model constructs (Hair et al., 2016). We see that those values are between 1 and 1,23 so below the critical value.

We measure the path significance through bias-corrected and accelerated bootstrapping. Since this is an exploratory study we as-



*** $p<0{,}001$, ** $p<0{,}05$, * $p<0{,}1$

Fig. 2. Structural model with Path coefficients and p values.

Table 7
Paths coefficients.

Paths	Orig. sample	Mean	St. Dev.	T	p
AV → CooT	0,492	0,498	0,066	7,474	0,000
CT → CooT	0,235	0,249	0,085	2,770	0,006
CooT → CPS	0,285	0,288	0,134	2,127	0,034

sumed a two-tailed test with a significance level of 10%, following (Hair et al., 2016). As we can see from Table 7, all indicators comply with their respective critical values. In particular T-statistics are above 1,96 for all paths and the p-values are below the reference level of 0,1 (for 10% significance) and also below the more conservative value of 0,1 (Hair et al., 2016). Therefore, we conclude that all paths in the model are significant. This supports all of our three hypotheses H_1, H_2, H_3.

Passing now to the evaluation of the R-square values of the two endogenous variables we see that Computational Thinking and Agile Values explain very well the construct Cooperative Thinking with a value of 0,374. Interestingly, Complex Problem Solving has a relatively low R-square value of 0,081 for two reasons (Hair et al., 2016).

The first one is statistical. Since CPS is derived by another endogenous construct, the statistical explanation power is mitigated by the mid-construct CooT. So, it is normal to have a relatively lower value.

The second reason is conceptual and regards the exploratory nature of this study. Although CooT is a useful skill for complex (or wicked) problems, since its p-value is significant, it may not be enough. With different words, there might be other constructs which address better complex problems. As well, there might be also more than just one construct which addresses complex problems. This remains an open issue, and future research should test new constructs (such as CPS) grounded in literature, which could represent a better fit in the model.

Going back to the validation of our findings we look now at the f-square values. These metrics indicates how well each exogenous construct explains the endogenous ones. Here we have that the relationship AV → CooT has the highest value of 0,35 which suggests a very high effect, according to literature standards (Hair et al., 2016). The relationship CT → CooT has a moderate effect, but still significant since it is above the threshold of 0,02 (Hair et al., 2016), with a value of 0,08. The same for the relationship CooT → CPS with a value of 0,09.

Now we to test the predictive validity of the model, to see if the exogenous constructs explains significantly the endogenous ones (Akter et al., 2011). To do so, we use run blindfolding with an omission distance of 7 to measure the Stone–Geisser's Q-square through Construct Crossvalidated Redundancy (Stone, 1974; Geisser, 1974). Here, the Q-square should be bigger than 0 (Hair et al., 2016). We have for both for CooT (Q^2: 0,177) and CPS (Q^2: 0,029) the match of this criterion.

We conclude that our structural model, represented in Fig. 2, can predict all tested constructs. Nevertheless, the low R^2 of CPS indicates that the model is not complete. Still, it is significant and is a solid ground to build new theory on. Therefore, we also conclude that CooT is a significant proxy to Complex Problem Solving. Significance and explanatory values of CooT suggest that AV and CT are good constructs for this new competence.

6. Discussion

Our structural model suggests a positive answer for both our Research Questions, according to these statistical considerations:

- RQ_1: the high R^2 of CooT indicates a high explanatory power of the new construct. This means that both CT and AV are significant components of this new overarching construct. Moreover, path coefficients of H_1 and H_2 are highly significant. So, they influence the new construct in a statistically significant way.
- RQ_2: CooT does explain in a significant way CPS, due to its path coefficient. Also H_3 is significant, considering the path's p- and absolute value. Since R^2 is not a relevant indicator in this case for the above-mentioned reasons, we can state that it does well explain CPS, and thus wicked problems.

From this evidence we can conclude that both CT and AV are building constructs of CooT, which is able to explain independently a new construct, namely Complex Problem Solving.

Consistently with our research design, we outline now the educational implications of this study and its limitations.

6.1. Implications

Our findings support the idea that CT and AV reinforce each other to sustain the new construct of Cooperative Thinking. Now we outline some educational practices that we included in our courses to foster Cooperative Thinking.

Firstly, it is useful to position Cooperative Thinking.

It is not a teaching method, like Project-Based Learning (PBL) (Blumenfeld et al., 1991). However, it is enhanced by teaching approaches which are student-centred and cooperative-based, like PBL or Problem-Based Learning (Savery and Duffy, 1995). At this stage of our research, we do not provide specific recommendations on didactic aspects, just content wise.

Cooperative Thinking is a competence, not a skill. Following the European Union's definition, a competence is the "*ability to use knowledge, skills and personal, social and/or methodological abilities, in work or study situations and in professional and personal development. It is not limited to cognitive elements (involving the use of theory, concepts or tacit knowledge); it also encompasses functional aspects (including technical skills) as well as interpersonal attributes (e.g. social or organizational skills) and ethical values*" (European Centre for the Development of Vocational Training, 2014). While, according to the same taxonomy, a skill is the

"*ability to apply knowledge and use know-how to compete tasks and solve problems*" (European Centre for the Development of Vocational Training, 2014). We stress this distinction (although it is often used as a synonym), since Cooperative Thinking is not task-specific, but it is traversal, encompassing both technical and social skills. In particular, to address CooT we suggest to develop specific skills and competences of both social and technical nature.

These activities are all linked to Cooperative Thinking (Missiroli et al., 2017b), which has also been used as baseline for our scale development in Section 4.3. Most of the proposed practices are also grounded in the pedagogical literature. Cooperative Thinking can be operationalized through established educational practices. The educational scope is to tackle key concepts of problem description, recognition, decomposition, in order to solve them computationally in teams, stressing cooperation and social sustainability. This reinforces the theoretical ground of this construct, since it is both backed in literature and is empirically significant. We stress the fact that using already mature practices is an effective way to support CooT, spreading this new competences in daily classes.

Students experience in an incremental way complex problems to learn reusable cooperation patterns. We propose (and have tested most of) the following categories of practices to foster CooT in everyday activities:

- *Complex negotiation*: when given a project problem, students are invited to discuss and evaluate alternative ideas and solutions, considering different viewpoints. Deriving from Agile negotiation (Beck and Andres, 2004) and negotiation pedagogy (Avruch, 2000), this aims to develop adequate capabilities to deal with stratified issues and different opinions. Finding a group-wise sustainable way to devise a solution of a problem, taking into consideration a variety of useful or useless points of view is the aim of this practice. Key activities include: structured brainstorming, architectural design and code contests, Randoris and Code retreats (Sato et al., 2008). For instance, we have developed specific exercises to let our students to develop and discuss the (mainly non functional) properties of a new product, explicitly asking them to analyze the trade-offs among the properties they believe that should be satisfied by the product.
- *Continuous learning*: this has to do with shaping a team to adapt to changes in the problem to solve. Both individuals and their groups should be ready to find and gain the knowledge needed to solve a given problem at hand. Education should be centered on enhancing the students' ability related to "reflection-in-action" (Schön, 1987), practicing continued learning and problem solving throughout their entire career. Activities such as Peer Learning and Exploratory learning are well suited to this task. An interesting exercise consists in working in pairs to a set of refactoring exercises driven by tests: the students have to learn how to exploit the different tests to evolve their code. An example we use is called Refactoring Golf and is available on GitHub.[2]
- *Group awareness*: this indicates the capability to be part of a group. It covers knowledge and perception of behavioral, cognitive, and social context information within a group (Bodemer and Dehler, 2011). It requires reflective activities (such as, Kristiansen and Rasmussen, 2014 Lego Serious Play, or Lego Scrum, Steghöfer et al., 2016; Steghöfer et al., 2017; Steghöfer, 2018) and group games in order to develop a "team spirit" and promote the self-organizing skill of the team. For instance, we ask the students to keep a diary of both individual and group activities, and to relate such artifacts to the shared board (or kanban) that is used by each group.
- *Group organization*: this refers to the ability to develop software as a group, i.e. deliver a working product collaboratively. This goal can be achieved by regularly applying Group-oriented Project-based learning, starting with small, toy project and scaling to complex ones. It is grounded within the domain of peer learning, to generate productive instructional dialogues for joint problem solving, relying on intrinsic rather than extrinsic rewards, discouraging competition between students (Damon and Phelps, 1989).
- *Social adaptability*: this refers to the groups' internal and external social dynamics. Especially for adolescents this kind of competence is a pivotal aspect of education; it will determine how future adults will be oriented to express social sensitivity (Adams, 1983). Activities include role play, group exercises, project simulations, and even stress tests, as in Kuhrmann and Münch (2016). A notable example here are entrepreneurial skills, since students are motivated to create value for stakeholders, being able to adapt themselves to a changing context (Burden et al., 2019).

The novelty of the CooT construct does not lie in the advancement of new skills, rather in the combination of different skills, encompassed in a new computer science-related competence. The result is the proposal of a new competence which aim is to support cooperative problem solving of technical contents.

6.2. Limitations

As inherent in any scientific method (Wohlin et al., 2012), this study has several limitations.

The first issue is about the use of cross-sectional data (i.e., observation of the population through data collection from many subjects at the same point of time) for the empirical assessment of the model. Hence, results may reflect associations rather than prediction between constructs. Moreover, it is not possible to assess if the relationship will change over time. However, a longitudinal study might overcome this limitation. Generally speaking, we tackled these issues through a sound theoretical derivation, which is a correct way to minimize these limitations (Hair et al., 2016).

Secondly, we measured our constructs from a subjective perspective through a single-informant approach. So, the constructs represent the students' perspective. Respondents may not have answered the question accurately or with some biases. For this reason the survey was anonymous and no grades were assigned for the participation at this research. Moreover, a sample size of 116 observations through different European countries minimized the method bias (Kim and Cavusgil, 2009). Third, we used perceptual measures, rather than objective ones, asking students to state their level of agreement on literature-derived items. So, the measurements may not fully reflect the real world accurately due to potential respondent bias and random errors. Therefore, items were adapted from previous studies and literature and subject to various examinations for ensuring their quality. However, continuous item development and validation is needed to update the constructs.

Finally, the last limitation regards the sampling technique. We used a stratified convenience sampling technique, were strata were defined accordingly to the acquired skill-set. We selected students who already had acquired in their curriculum both training and experience with CT and AV exogenous constructs. This enabled us to assess the level of endogeneity of CooT and CPS. In doing so, we asked European partner Universities we already collaborate with to administer the survey. Those Universities adopted curricula that

[2] https://github.com/sf105/refactoring-golf.

fostered CT and AV and were therefore considered suitable targets for our strata definition. Our research did not target non-European educational environments; this may weaken our results, since cultural factors may have also played a role, which we did not consider in this study. Generally, non-responses may have lead to sample selection bias if a systematic and unobservable difference exists between respondents and non-respondents (Whitehead et al., 1993).

All in all, we consider our limitations acceptable for this exploratory study, especially because we took several precautions to minimize them. As discussed in Section 5, all statistical indicators suggest the conceptual validity of the model. Still, we are aware that this is a starting point, not an ending one; further research is needed to generalize the model and to better define its sub-dimensions.

7. Conclusions

With this paper we validated the theoretical model of Cooperative Thinking to train teams of students to manage software engineering problems. Accordingly, we are advancing a new computer science competence which aim is to support cooperative problem solving of technical contents to address complex software engineering problems. We defined Cooperative Thinking as a competence encompassed by Complex Negotiation, Continuous Learning, Group Awareness, and Group Organization, and explained how we have used them in class.

To validate the proposed educational model we used Structural Equation Modeling with Partial Least Squares. Exploiting this technique we were able to test the statistical significance of the relationships between constructs as also their explanatory power. Indeed, PLS-SEM has important potentials in software engineering to test the significance of theoretical social constructs.

This study provided a model for our future empirical investigations on the new educational construct. Our future work will focus on both theoretical and pedagogical aspects.

Some generalization efforts need to be undertaken to consider Cooperative Thinking as a real universal competence. This study could be administered also in non-European countries. To uncover unobserved heterogeneity in the inner (structural) model a Finite Mixture Partial Least Squares (FIMIX-PLS) segmentation test should be run (Hahn et al., 2002). This will capture heterogeneity by estimating the probabilities of segment memberships for each observation and simultaneously estimate the path coefficients of all segments. Doing so, an improved understanding of constructs performance on different segments (i.e., groups of students) is possible. Thus, it is possible to tailor educational curricula, according to each segments' sensibility, according to respective differences (e.g., performance, culture, gender, age, students' level). Moreover, this can be supported by finer granular studies, based on students' composition, to analyze those pedagogical differences. Literature work is further needed to refine measurements and sub-dimensions of all constructs. We used Complex Problem Solving as a proxy of wicked problem. However, this assumption needs further insights to be validated. In a possible extension of the model, wicked problems may be represented by other parent-constructs of CPS to make their representation more trustworthy.

From a pedagogical perspective, Cooperative Thinking practices and educational curricula need to be outlined in more depth with respect to what we did in this paper. Indeed, it is possible that an *ad hoc* curriculum on Cooperative Thinking will help students to improve model fitting. For instance, developing the proposed constructs of Complex Negotiation, Continuous Learning, Group Awareness, and Group Organization.

Data

Raw data and calculations are openly available under CC BY 4.0 license at DOI: 10.6084/m9.figshare.7127069.

Acknowledgments

The authors thank all the colleagues of the universities of Bologna, Chalmers, Innopolis, Limerick, Modena, and the High School IIS F. Corni who helped us spreading the survey; in particular Tiziana Margaria, Jan-Philipp Steghöfer, Giacomo Cabri, Carlo Eutropio as well as the students who carefully answered.

This work was partially supported by the Institute of Cognitive Sciences and Technologies of the Italian National Research Council (ISTC-CNR); the Italian Inter-University Consortium for Informatics (CINI); and the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero – the Irish Software Research Centre (www.lero.ie).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jss.2019.110401.

References

Adams, G., 1983. Social competence during adolescence: social sensitivity, locus of control, empathy, and peer popularity. J. Youth Adolesc. 12 (3), 203–211.
Akter, S., D'Ambra, J., Ray, P., 2011. An evaluation of PLS based complex models: the roles of power analysis, predictive relevance and GoF index. In: Proceedings of the Americas Conference on Information Systems, pp. 1–7.
Allert, J., 2004. Learning style and factors contributing to success in an introductory computer science course. In: Proceedings of the International Conference on Advanced Learning Technologies. IEEE, pp. 385–389.
Alliance, A., 2001. Agile manifesto. Online at http://www.agilemanifesto.org 6.
Avruch, K., 2000. Culture and negotiation pedagogy. Negot. J. 16 (4), 339–346.
Barr, V., Stephenson, C., 2011. Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? ACM Inroads 2 (1), 48–54.
Barrick, M.R., Stewart, G.L., Neubert, M.J., Mount, M.K., 1998. Relating member ability and personality to work-team processes and team effectiveness. J. Appl. Psychol. 83 (3), 377–391.
Batra, D., 2018. Agile values or plan-driven aspects: which factor contributes more toward the success of data warehousing, business intelligence, and analytics project development? J. Syst. Softw. 146, 249–262.
Beck, K., Andres, C., 2004. Extreme Programming Explained: Embrace Change. Addison–Wesley.
Blackwell, A., Church, L., Green, T., 2008. The abstract is' an enemy': alternative perspectives to computational thinking. In: Proceedings of the PPIG, 8, pp. 34–43.
Blumenfeld, P., et al., 1991. Motivating project-based learning: sustaining the doing, supporting the learning. Educ. Psychol. 26 (3-4), 369–398.
Bodemer, D., Dehler, J., 2011. Group awareness in CSCL environments. Comput. Hum. Behav. 27 (3), 1043–1045.
Buchanan, R., 1992. Wicked problems in design thinking. Des. Issues 8 (2), 5–21.
Burden, H., Steghöfer, J.-P., Svensson, O.H., 2019. Facilitating entrepreneurial experiences through a software engineering project course. In: Proceedings of the International Conference on Software Engineering Companion - SEET. ACM, p. preprint.
Camillus, J.C., 2008. Strategy as a wicked problem. Harv. Bus. Rev. 86 (5), 98.
Carroll, J.M., 1990. The copernican plan: restructuring the American high school. Phi Delta Kappan 71 (5), 358–365.
Carter, L., 2011. Ideas for adding soft skills education to service learning and capstone courses for computer science students. In: Proceedings of the Technical Symposium on Computer Science Education. ACM, pp. 517–522.
Chin, W., R. Newsted, P., 1996. Structural Equation Modeling Analysis with Small Samples Using Partial Least Square. Sage.
Chin, W.W., 1998a. Commentary: Issues and Opinion on Structural Equation Modeling. MIS Quarterly 22 (1), vii–xvi. Retrieved from http://www.jstor.org/stable/249674.
Chin, W.W., 1998b. The partial least squares approach to structural equation modeling. Mod. Methods Bus. Res. 295 (2), 295–336.
Conway, M., 1968. How do committees invent. Datamation 14 (4), 28–31.
Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., Woollard, J., 2015. Computational thinking: a guide for teachers.
Damon, W., Phelps, E., 1989. Critical distinctions among three approaches to peer education. Int. J. Educ. Res. 13 (1), 9–19.
Denning, P.J., 2017. Remaining trouble spots with computational thinking. Commun. ACM 60 (6), 33–39.

Diamantopoulos, A., Siguaw, J.A., 2006. Formative versus reflective indicators in organizational measure development: a comparison and empirical illustration. Br. J. Manag. 17 (4), 263–282.

Dibbern, J., Goles, T., Hirschheim, R., Jayatilaka, B., 2004. Information systems outsourcing: a survey and analysis of the literature. ACM Sigmis Database 35 (4), 6–102.

Dijkstra, T.K., Henseler, J., 2015. Consistent and asymptotically normal PLS estimators for linear structural equations. Comput. Stat. Data Anal. 81, 10–23.

Dingsøyr, T., Lassenius, C., 2016. Emerging themes in agile software development: introduction to the special section on continuous value delivery. Inf. Softw. Technol. 77, 56–60.

Durak, H., Saritepeci, M., 2018. Analysis of the relation between computational thinking skills and various variables with the structural equation model. Comput. Educ. 116, 191–202.

EC, 2017. Key competences for lifelong learning: European reference framework. http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=LEGISSUM:c1109.

Edwards, J.R., Bagozzi, R.P., 2000. On the nature and direction of relationships between constructs and measures.. Psychol. Methods 5 (2), 155.

Efron, B., Tibshirani, R.J., 1994. An Introduction to the Bootstrap. CRC Press.

European Centre for the Development of Vocational Training, 2014. Terminology of European Education and Training Policy: A Selection of 130 Key Terms. Office for Official Publications of the Europ. Communities.

Faul, F., Erdfelder, E., Buchner, A., Lang, A., 2009. Statistical power analyses using g* power 3.1: tests for correlation and regression analyses. Behav. Res. Methods 41 (4), 1149–1160.

Fornell, C., Larcker, D.F., 1981. Evaluating structural equation models with unobservable variables and measurement error. J. Mark. Res. 18 (1), 39–50. doi:10.2307/3151312.

Fraj-Andrés, E., Lucia-Palacios, L., Pérez-López, R., 2018. How extroversion affects student attitude toward the combined use of a wiki and video recording of group presentations. Comput. Educ. 119, 31–43.

Gefen, D., Straub, D., Boudreau, M.-C., 2000. Structural equation modeling and regression: guidelines for research practice. Commun. AIS 4 (1), 7.

Geisser, S., 1974. A predictive approach to the random effect model. Biometrika 61 (1), 101–107.

Goggins, S., Xing, W., 2016. Building models explaining student participation behavior in asynchronous online discussion. Comput. Educ. 94, 241–251.

Gudergan, S.P., Ringle, C.M., Wende, S., Will, A., 2008. Confirmatory tetrad analysis in PLS path modeling. J. Bus. Res. 61 (12), 1238–1249.

Hahn, C., Johnson, M.D., Herrmann, A., Huber, F., et al., 2002. Capturing customer heterogeneity using a finite mixture PLS approach. Schmalenbach Bus. Rev. 54 (3), 243–269.

Hair, J.F., Hult, G.T., Ringle, C., Sarstedt, M., 2016. A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM). Sage Publications.

Hair, J.F., Ringle, C.M., Sarstedt, M., 2011. PLS-SEM: indeed a silver bullet. J. Mark. Theory Pract. 19 (2), 139–152.

Henderson, P.B., 2009. Ubiquitous computational thinking. Computer 42 (10), 100–102. doi:10.1109/MC.2009.334, (October 2009).

Henseler, J., Ringle, C.M., Sarstedt, M., 2015. A new criterion for assessing discriminant validity in variance-based structural equation modeling. J. Acad. Mark. Sci. 43 (1), 115–135.

Henseler, J., Ringle, C.M., Sinkovics, R.R., 2009. The use of partial least squares path modeling in international marketing. In: New Challenges to International Marketing. Emerald Group Publishing Limited, pp. 277–319.

Higgins, C.A., Duxbury, L.E., Irving, R.H., 1992. Work-family conflict in the dual-career family. Organ. Behav. Hum. Decis. Process. 51 (1), 51–75.

Hoskey, A., Zhang, S., 2017. Computational thinking: what does it really mean for the k-16 computer science education community. J. Comput. Sci. Coll. 32 (3), 129–135.

Howard, R.A., Carver, C.A., Lane, W.D., 1996. Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: tying it all together in the CS2 course. ACM SIGCSE Bull. 28, 227–231. ACM.

Hulland, J., 1999. Use of partial least squares (PLS) in strategic management research: a review of four recent studies. Strateg. Manag. J. 20 (2), 195–204. Retrieved from http://www.jstor.org/stable/3094025.

Hung, W., Jonassen, D.H., Liu, R., et al., 2008. Problem-based learning. In: Handbook of Research on Educational Communications and Technology, vol. 3. Routledge, pp. 485–506.

Johnson, D., et al., 1994. Cooperative Learning in the Classroom. ERIC.

Katz, D.L., 1960. Conference report on the use of computers in engineering classroom instruction. Commun. ACM 3 (10), 522–527.

Kim, D., Cavusgil, E., 2009. The impact of supply chain integration on brand equity. J. Bus. Ind. Mark. 24 (7), 496–505.

Kirschner, P., Sweller, J., Clark, R., 2006. Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. Educ. Psychol. 41 (2), 75–86.

Korkmaz, O., Cakir, R., YasarOzden, M., 2017. A validity and reliability study of the computational thinking scales (CTS). Comput. Hum. Behav. 72, 558–569.

Kristiansen, P., Rasmussen, R., 2014. Building a Better Business Using the Lego Serious Play Method. John Wiley & Sons.

Kropp, M., Meier, A., 2013. Teaching agile software development at university level: values, management, and craftsmanship. In: Proceedings of the Conference on Software Engineering Education and Training. IEEE, pp. 179–188.

Kropp, M., Meier, A., 2014. New sustainable teaching approaches in software engineering education. In: Proceedings of the Global Engineering Education Conference. IEEE, pp. 1019–1022.

Kuhrmann, M., Münch, J., 2016. When teams go crazy: an environment to experience group dynamics in software project management courses. In: Proceedings of the International Conference on Software Engineering. ACM, pp. 412–421.

Lasserre, P., Szostak, C., 2011. Effects of team-based learning on a cs1 course. In: Proceedings of the Annual Joint Conference on Innovation and Technology in Computer Science Education. ACM, pp. 133–137.

Lindsjørn, Y., Sjøberg, D.I., Dingsøyr, T., Bergersen, G.R., Dybå, T., 2016. Teamwork quality and project success in software development: a survey of agile development teams. J. Syst. Softw. 122, 274–286.

Lu, E.Y., Ma, H., Turner, S., Huang, W., 2007. Wireless internet and student-centered learning: a partial least-squares model. Comput. Educ. 49 (2), 530–544.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E., 2010. The scratch programming language and environment. ACM Trans. Comput. Educ. 10 (4), 16.

Meerbaum-Salant, O., Hazzan, O., 2010. An agile constructionist mentoring methodology for software projects in the high school. ACM Trans. Comput. Educ. 9 (4), 29. doi:10.1145/1656255.1656259, Article 21 (January 2010).

Meier, A., Kropp, M., Perellano, G., 2016. Experience report of teaching agile collaboration and values: agile software development in large student teams. In: Proceedings of the International Conference on Software Engineering Education and Training. IEEE, pp. 76–80.

Michaelsen, L.K., Knight, A.B., Fink, L.D., 2002. Team-Based Learning: A Transformative Use of Small Groups. Greenwood Publishing Group.

Missiroli, M., Russo, D., Ciancarini, P., 2016a. Learning agile software development in high school: an investigation. In: Proceedings of the International Conference on Software Engineering. ACM, pp. 293–302.

Missiroli, M., Russo, D., Ciancarini, P., 2016b. Una didattica agile per la programmazione. Mondo Digitale 15 (64).

Missiroli, M., Russo, D., Ciancarini, P., 2017a. Agile for millennials: a comparative study. In: Proceedings of the 1st International Workshop on Software Engineering Curricula for Millennials. IEEE, pp. 47–53.

Missiroli, M., Russo, D., Ciancarini, P., 2017b. Cooperative thinking, or: computational thinking meets agile. In: Proceedings of the Conference on Software Engineering Education and Training. IEEE, pp. 187–191.

Montori, F., Bedogni, L., Bononi, L., 2018. A collaborative internet of things architecture for smart cities and environmental monitoring. IEEE Internet Things J. 5 (2), 592–605.

Nagel, T., 1986. The View from Nowhere. Oxford University Press.

Nunnally, J., 1978. Psychometric Methods. McGraw-Hill.

Papert, S., 1980. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc.

Papert, S., Harel, I., 1991. Situating Constructionism, 36. Ablex Publishing Corporation.

Polya, G., 1957. How to Solve it: A New Aspect of Mathematical Method. Princeton University Press.

Popper, K., 2005. The Logic of Scientific Discovery. Routledge.

Raskino, M., Waller, G., 2016. Digital to the Core: Remastering Leadership for your Industry, Your Enterprise, and Yourself. Routledge.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al., 2009. Scratch: programming for all. Commun. ACM 52 (11), 60–67.

Ringle, C.M., Sarstedt, M., Straub, D., 2012. A critical look at the use of PLS-SEM in MIS quarterly. Management Information Systems Quarterly 36 (1), iii–xiv.

Ringle, C.M., Wende, S., Becker, J.-M., 2015. Smartpls 3. Boenningstedt: SmartPLS GmbH.

Rittel, H., Webber, M.M., 1973. 2.3 planning problems are wicked. Polity 4, 155–169.

Rivera-Ibarra, J.G., Rodríguez-Jacobo, J., Serrano-Vargas, M.A., 2010. Competency framework for software engineers. In: Proceedings of the International Conference on Software Engineering Education and Training. IEEE, pp. 33–40.

Roman-Gonzalez, M., et al., 2018. Extending the nomological network of computational thinking with non-cognitive factors. Comput. Hum. Behav. 80, 441–459.

Russo, D., Missiroli, M., Ciancarini, P., 2018. A conceptual model for cooperative thinking. In: Proceedings of the International Conference on Software Engineering Companion. ACM.

Russo, D., Stol, K.-J., 2019. Soft theory: a pragmatic alternative to conduct quantitative empirical studies. In: Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice (CESSER-IP '19). IEEE Press, Piscataway, NJ, USA, pp. 30–33. doi:10.1109/CESSER-IP.2019.00013.

Sato, D.T., Corbucci, H., Bravo, M.V., 2008. Coding dojo: an environment for learning and sharing agile practices. In: Proceedings of the Agile Conference. IEEE, pp. 459–464.

Savery, J.R., Duffy, T.M., 1995. Problem based learning: an instructional model and its constructivist framework. Educ. Technol. 35 (5), 31–38.

Schön, D., 1987. Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions. Jossey-Bass.

Seman, L.O., Hausmann, R., Bezerra, E.A., 2018. On the students' perceptions of the knowledge formation when submitted to a project-based learning environment using web applications. Comput. Educ. 117, 16–30.

Shakroum, M., Wong, K.W., Fung, C.C., 2018. The influence of gesture-based learning system (GBLS) on learning outcomes. Comput. Educ. 117, 75–101.

Slife, B., Williams, R., 1995. What's Behind the Research?: Discovering Hidden Assumptions in the Behavioral Sciences. Sage.
Steghöfer, J., 2018. Providing a baseline in software process improvement education with lego scrum simulations. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp. 126–135.
Steghöfer, J.-P., Burden, H., Alahyari, H., Haneberg, D., 2017. No silver brick: opportunities and limitations of teaching scrum with lego workshops. J. Syst. Softw. 131, 230–247.
Steghöfer, J.-P., Knauss, E., Alégroth, E., Hammouda, I., Burden, H., Ericsson, M., 2016. Teaching agile: addressing the conflict between project delivery and application of agile methods. In: Proceedings of the International Conference on Software Engineering Companion. ACM, pp. 303–312.
Steghöfer, J.-P., et al., 2018. Involving external stakeholders in project courses. ACM Trans. Comput. Educ. 18 (2), 8:1–8:32.
Stone, M., 1974. Cross-validatory choice and assessment of statistical predictions. J. R. Stat. Soc. Ser. B (Methodological) 36 (2), 111–147. Retrieved from http://www.jstor.org/stable/2984809.
Sweller, J., 2004. Instructional design consequences of an analogy between evolution by natural selection and human cognitive architecture. Instr. Sci. 32 (1-2), 9–31.
Thomas, L., Ratcliffe, M., Woodbury, J., Jarman, E., 2002. Learning styles and performance in the introductory programming sequence. ACM SIGCSE Bull. 34, 33–37. ACM.
Trilling, B., Fadel, C., 2012. 21st Century Skills: Learning for Life in Our Times. John Wiley & Sons.
Vv.Aa., 2011. Operational definition of computational thinking by the ACM computer science teachers association. http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf.
Vv.Aa., 2015. Computational thinking: a guide for teachers by the british computer society. http://community.computingatschool.org.uk/files/6695/original.pdf.
Vv.Aa., 2016a. 21st century skills by the glossary of education reform. http://edglossary.org/21st-century-skills/.
Vv.Aa., 2016b. ISTE standards for students by the international society for technology in education. http://www.iste.org/standards/standards/for-students-2016.
Weber, E.P., Khademian, A.M., 2008. Wicked problems, knowledge challenges, and collaborative capacity builders in network settings. Public Adm. Rev. 68 (2), 334–349.
WEF, 2016. The future of jobs: employment, skills and workforce strategy for the fourth industrial revolution. http://www3.weforum.org/docs/WEF_Future_of_Jobs.pdf.
Werts, C.E., Linn, R.L., Jöreskog, K.G., 1974. Intraclass reliability estimates: testing structural assumptions. Educ. Psychol. Meas. 34 (1), 25–33.
Whitehead, J.C., Groothuis, P.A., Blomquist, G.C., 1993. Testing for non-response and sample selection bias in contingent valuation: analysis of a combination phone/mail survey. Econ. Lett. 41 (2), 215–220.
Wing, J., 2006. Computational thinking. Commun. ACM 49 (3), 33–35.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer Science & Business Media.
Wold, H., 1974. Causal flows with latent variables: partings of the ways in the light of NIPALS modelling. Eur. Econ. Revi. 5 (1), 67–86.
Wold, H., 1983. Systems analysis by partial least squares.
Yeh, R.T., 1991. System development as a wicked problem. Int. J. Softw. Eng. Knowl. Eng. 1 (02), 117–130.

Paolo Ciancarini is Professor of Computer Science at the Universities of Bologna and Innopolis (adjunct professor); he has a PhD in Computer Science from the University of Pisa.

Marcello Missiroli is postdoc of Computer Science at the University of Bologna; he has a PhD in Computer Engineering from the University of Modena and ReggioEmilia.

Daniel Russo is Post Doctoral Researcher in Computer Science at Lero & University College Cork; he has a PhD in Computer Science & Engineering from the University of Bologna.