ISSN 1392-1126. eISSN 2424-6158. 2018 Priedas
DOI: https://doi.org/10.15388/Problemos.2018.0.12346

COMPUTATIONAL PRACTICES, EDUCATIONAL THEORIES, AND LEARNING DEVELOPMENT

Don Passey
Lancaster University
d.passey@lancaster.ac.uk

Valentina Dagienė
Institute of Educational Sciences
Vilnius University
valentina.dagiene@mii.vu.lt

Loice Victorine Atieno
Eötvös Loránd University
atienomunira04@gmail.com

Wilfried Baumann
Austrian Computer Society
baumann@ocg.at

Abstract. *Many countries are adopting computing (or informatics) in schools, for pupils from 5 years of age. Educational philosophies (and learning theories) that such curricula might be based on are not clear in curriculum documentation. Many Western countries' curricula are based on developmental concepts of cognitive constructivism, with activities progressing through sensorimotor, preoperational, concrete operational, and formal operational stages. Social constructivism and constructionism add new dimensions to this learning framework, both fundamentally important for developing computing practices. We review selected learning theories, and investigate features that should underpin computing curricula if practices and outcomes are to develop computing practitioner competencies of a software developer.*

Keywords: *computational practices; educational philosophy; learning theories; learning progression; software developer competencies*

Background

Many countries are adopting, or have recently adopted, computing as a school subject or discipline (such as the national curriculum for computing in England, 2013), although the subject may have alternative names and somewhat different concerns in different countries (computer science in the United States or Canada, or informatics in Germany, Poland or Lithuania, for example). Computing learning practices in these 'new' curricula can involve pupils from 5 years of age, perhaps within a discrete subject, or perhaps integrated into subject topics across the curriculum. A key question to ask is – what educational philosophies or theories underpin practices

and planning of teachers and educators when developing appropriate curricula and lessons? Learning concepts, frameworks or theories that such curricula are based on, which therefore underpin practices and outcomes across the age span of learners, are not clear in curriculum documentation. It is clear that educational philosophies need to address two dimensions: depth of learning experience (the quality of learning activities and interventions), and breadth or length of learning experience (the progression of activities and interventions across a learner's educational career).

Whether a single educational philosophy will provide for a teacher's planning and development needs is not clear. Western curriculum concepts of learning progression are largely based on Jean Piaget's research (1936), describing learning as a form of cognitive constructivism, developing over the age span of young people, progressing through a series of stages or phases: sensorimotor, preoperational, concrete operational, and formal operational. While Lev Vygotsky's research (1978) added a more social dimension of learning, a concept of social constructivism, this did not necessarily add to the concept of progression. With the advent of digital and computing-based resources, further concepts of learning have been developed. From the perspective of computing, the most significant of these is perhaps the concept of constructionism developed from Seymour Papert's research (1986). While this research and its resultant theory added to the concept of depth and quality of learning activities, it did not substantially add to the concept of progression. In summary, there has been an emergence of learning theories over the past 80 years, tied in certain ways to developments and uses of digital resources in education and society. Our concepts of learning have been reconsidered and regenerated from a cognitive individual perspective to a social perspective with others, and then with digital resources. Our current contexts, however, which now include wide uses of mobile devices and social media, for example, are quite different from those contexts in which researchers including Piaget, Vygotsky and Papert developed their learning concepts or theories. Does this mean that we need to reconsider educational theories in our current context? This is particularly pertinent, since the implementation of computing into school curricula is developing interests in educators around the concept of computational thinking (the thinking required in order to develop computing competencies (Wing 2008), from school to work-based professional practices).

This paper explores the following issue: what educational (learning) theories should underpin computing curricula to ensure that short- and longer-term professional practices and outcomes are appropriate and effective across the age span of learners.

Research questions

The research questions we ask are:

- Can current educational (learning) theories be reliably applied to develop computational practices across the age range from 5 to 18 years?
- Do we need a new conception of learning that accommodates our current context?
- Can we create this concept or theory from a research perspective, or can we do it from practical know-how and experience?

- What should we do in the future to underpin learning development in computing?

Approach

We explore the research questions by taking a systematic analytic approach as follows.

To develop breadth and length of learning experience, we need to identify what our end-point will be. One key reason for including computing education in schools is to support engagement with longer-term interest and development leading to professional practice. For this analysis, therefore, we describe and detail competencies required of a contemporary computer professional, a software developer, as the end-point. We could alternatively or additionally include competencies of other computing specialists, such as network or software engineers, but we use the competencies of a software developer to illustrate our approach and concerns.

We then take a number of learning theories, selected as being relevant to our analysis, describing and detailing their sources, and their main features. Other learning theories are not included, but at this stage of analysis, we wish to exemplify our approach and concerns. As educational concepts or practices (concerning teaching as well as learning) are derived from learning theories rather than being developed as standalone theories, we focus in this paper on learning theories rather than on educational theories.

Using features of each learning theory, we identify whether and to what extent there is a considered match with the practices, competencies required of a contemporary software developer, and whether certain competencies would not be easily developed if teachers and educators used each specific learning theory to underpin educational practices.

We critically map the findings from across the learning theories explored, indicating where there are matches and where there are gaps, and considering whether a new learning theory is required, and how this relates to previous learning theories.

From the outcomes of the mapping, we identify the rationale for, a possible name for, and detail the nature of a new learning theory proposed.

This is an ambitious project. Hence, we do not provide in this paper a picture that cannot be further developed and questioned. It is our intention to open up the research questions, which we see as being important strategically for the future of computing (informatics) development in education. Learning theories continue to evolve in parallel with social constructs; we see this paper as a contribution within that evolution.

Competencies Required of a Contemporary Software Developer or Computer Programmer

Competencies (which also in this paper include skills, as described and defined by other authors) needed by a software developer can be detailed through nine different elements or processes (identified in, for example, Ahmed, et al. 2013; Surakka and Malmi 2004):

- Conceiving – taking user, market, technical and end-product requirements into consideration, producing ideas or drafts of a software process or product.
- Planning – exploring scope and defining specific user requirement, outlining and

detailing the overall process and time plan leading to a final outcome.

- Designing – generating an overview or high-level design of the outcome, identifying specific elements or modules of a program, how they integrate, and what language, operating system and hardware components might be involved.
- Developing – prototyping to consider proof-of-concept or trials of possible alternatives, leading to implementation that involves programming the code.
- Documenting – detailing the internal design, so that the software and process can be reviewed, revised and maintained in the future.
- Debugging/testing – finding and resolving problems that stop efficient and effective use of the software, and judging the quality of the outcome compared to initial requirements.
- Deploying – releasing the software for use, with additional concern for customisation.
- Evaluating/improving – beyond the initial deployment period, taking ongoing feedback into account, and exploring how to improve, perhaps integrating further or new software facility.
- Maintaining software – picking up on problems or issues over longer periods of time, and exploring ways to resolve these within contemporary situations.

Competencies can be categorised in a number of different ways. Stephen Lamb, Quentin Maire and Esther Doecke (2017) categorised 21st century skills as: critical thinking; creativity; metacognition; problem solving; collaboration; motivation; self-efficacy; conscientiousness; and grit or perseverance. This categorisation focuses on what might be regarded as "soft skills", without emphasising skills regarded as operational or technical (manipulative and mechanical, for example). SkillScan (2012) takes an alternative approach, categorising competencies as: relationship; communication; management/leadership; analytical; creative; and physical or technical. For this paper, the competencies are broadly categorised into three different groups, formed from a coalescence of these sources: technical competencies (having a good understanding of the theoretical background of the field as well as common practices used and discussed by the community, including those enabling effective interactions between the individual with physical objects such as machines and technological systems); analytical competencies (abilities to collect and analyse information, problem-solve, and make decisions concerned with logical thinking, mathematical reasoning, and structural reasoning and planning); social and emotional competencies (effective communication skills when communicating with team members, customers, or team leaders, writing and understanding documentation such as manuals, tutorials, or trouble-shooting guides, being willing and able to share knowledge and expertise with fellow team members, working independently and within groups, analysing user needs and having a solid understanding of a company's needs, scrutinising information given, anticipating events that are hard to predict, and handling pressure and failure).

Some competencies listed above are more obviously critical requirements, often easier to assess. Some are more related to short-term success, and those would be expected to be in high demand in job

Table 1. ***Framework of required competencies of contemporary software developers.***

Elements or processes	Technical competencies	Analytical competencies	Social and emotional competencies
Conceiving			
Planning			
Designing			
Developing			
Documenting			
Debugging/testing			
Deploying			
Evaluating/improving			
Maintaining			

advertisements. Some competencies are more subtle, and related more to long-term success. Interestingly, these competencies relate strongly to those identified within school computing curricula. How school practices enable development towards these competency goals is our concern in this paper.

Taking the range of required competencies (considered in the next section of the paper), our framework for analysis is shown in Table 1. Using this framework, we consider whether each selected learning theory can underpin the creation of learning activities required to develop each of these competency sets. We answer the question: is it possible to understand how this learning theory provides for educational practice, relating to or explaining or underpinning the progression of learning activities to support each of these competencies?

Matching Learning Theories to Competencies of Contemporary Software Developers

Here, we take a number of learning theories, chosen because they either often underpin educational curricula and practices, or are associated in certain ways with computing (informatics) education and longer-term professional practices. For each learning theory, we consider their source(s), main features, the match to the development of learning activities to support the competencies of a contemporary software developer (the competencies identified in Table 1), and highlight competencies not clearly supported using this learning theory approach.

Cognitive constructivism

Source. Cognitive constructivism emerges from Piaget's studies on cognitive development (Piaget 1936,1952). He did not examine the development of learning per se, but on learning development from early learners up to 11 years of age (beyond this age, the development identified was considered to go across the lifespan). He gathered evidence from a small, discrete sample (3 of his own children, and some of his colleague's children in his later work).

Main features. Piaget's theory identifies three components of development: schemata (building blocks in memory that develop

Table 2. ***Framework of competencies related to cognitive constructivism.***

Elements or processes	Technical ompetencies	Analytical competencies	Social and emotional competencies
Conceiving	√ from 2 years of age	√ from 11 years of age	?
Planning	√ from 11 years of age	√ from 11 years of age	?
Designing	√ from 7 years of age	√ from 11 years of age	?
Developing	√ from 7 years of age	√ from 11 years of age	?
Documenting	√ from 7 years of age	√ from 11 years of age	?
Debugging/testing	√ from 11 years of age	√ from 11 years of age	?
Deploying	√ from 7 years of age	√ from 11 years of age	?
Evaluating/improving	√ from 11 years of age	√ from 11 years of age	?
Maintaining	√ from 11 years of age	√ from 11 years of age	?

arenas of knowledge or understanding); adaptation (how an individual moves from one stage of development to another); and stages of cognitive development. He described schemata as sequences of actions held in memory, providing blocks for the retention and building of knowledge, ideas and understanding. He described adaptation in three stages: assimilation (using an existing schema to explain or work with a new idea or piece of knowledge); accommodation (realisation that an existing schema does not apply and needs to be adapted); equilibration (when a schema needs to be replaced, and the challenges associated with doing this). He described overall stages of development as: sensorimotor (up to 2 years, when mental representations are held in memory or mind); preoperational (2 to 7 years, when an object or word is symbolically associated with something else); concrete operational (7 to 11 years, when internal thinking occurs without having to do this physically); and formal operation (from 11 years, when thinking is logical and about abstract concepts). Whether stages do exist, and what the influences of social and cultural factors might be, were not explored within his research. The role of language, and the concept of schemata, have both been questioned by other researchers.

Match to competencies of a contemporary software developer. Learning activities could be devised to support the development of competencies in the Table 1 framework, as shown in Table 2.

Competencies not covered. Taking Piaget's stages of development as a guide, some technical competencies could be developed from 2 years of age, while others could be developed from 7 years of age. If all the competencies could not be developed until 11 years of age, then this places a wide number and range of learning activities out of the central context of contemporary practice. Additionally, the stages of development do not allow a clear understanding of how necessary social and emotional competencies might be developed.

Social constructivism

Source. Social constructivism is concerned with the social construction of knowledge. Influenced by Vygotsky's work, it is centred on the social context of learning, recognising that knowledge is collectively created and assembled (Bodrova and Leong

2012; Gauvain 2008). Through interaction, learners can express their thoughts, hence creating a common understanding associated with an idea (Kalpana 2014). Margaret Gredler (2008) believes that in moving from the theory developed by Piaget towards that of Vygotsky, there is a shift of ideas from individualism to collaboration, aided performance, social communication and sociocultural activity. Accordingly, learning can be socially constructed, perhaps modelled, making a sequence of ideas public, for discussion and change. This practice parallels the needs of programming activity. Indeed, Papert indicated relationships between professional programmer work and children's programming actions.

Main features. Conversation is the most important means of maintaining, modifying and reconstructing subjective reality (according to Berger and Luckmann 1991). Vygotsky's theory is based on three major themes: social interaction, the "More Knowledgeable Other" (MKO) and the "Zone of Proximal Development" (ZDP). Social interaction enables the process of cognitive development, with learners exhibiting both social and individualistic functions. Learning through a MKO concerns learning through someone more familiar with the subject under study, whether a teacher, coach, older or younger person, peer, or even a computer. The ZPD is the cognitive gap or difference between the learner's ability to perform a task with the help of another or through collaboration and the time the learner performs the task independently.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of all competencies shown in Table 1, but the progression of these competencies is not clear from this learning theory.

Competencies not covered. Whilst it is possible for all competencies to be developed using this learning theory as a background, roles of different social actors will clearly be important. This will also relate to the progression of competencies and the development of appropriate learning activities. How teachers and significant others are involved, including the extent and forms of interactions, could easily determine the feasibility of any of these forms of competency development.

Social learning

Source. Introduced by Albert Bandura (1977), this theory sought to clarify how youngsters learn in social conditions, watching and afterwards mimicking the conduct of others. The theory postulates that individuals learn through observation, simulation, and demonstration. The theory has been described as a link between behaviourist and cognitive learning theories as it incorporates attention, memory, and motivation. The outcome of a learner's conduct is seen to result more from watched behaviour than one they had adopted themselves.

Main features. Social learning has four elements: observational learning, reciprocal determinism, self-regulation, and self-efficacy. Observational learning involves observing a model in action (a person, a description in a book or even a movie) and then imitating the same. For the model to be enacted, it must be given attention, what is observed is retained, and learners are motivated to apply what

has been learnt. Reciprocal determinism describes the influence of the environment, concerned with social interaction between the learner and others. Self-regulation involves the setting of goals, discipline for action, and follow-through for learning to occur and performance to improve. Lastly, self-efficacy involves the belief the learner has in their capability to learn and perform. Hence, creation of an environment that fosters confidence in the learner should be encouraged and cultivated.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of all competencies shown in Table 1, but the range of activities would be limited, and progression of these competencies is not clear from this learning theory.

Competencies not covered. Concerns with using this background learning theory are similar to those posed in discussing social constructivism. Although all competencies could be developed, the range and order of these, and the importance of developing and subsequent reliance upon metacognitive skills such as self-regulation, could affect the relationship of the development of these competencies.

Constructionism

Source. This educational development theory, developed by Papert, highlights that though learning happens inside a learner's mind, this happens when the person is engaged in a personally meaningful activity outside of their mind. Four aspects are essential to the design of constructionist learning environments: learning through designing, personalising, sharing, and reflecting. Constructionism is grounded in the belief that the most effective learning experiences grow out of active construction of all types of artefacts (Papert 1986), so it strongly resonates with the current maker movement (Martinez and Stager 2013). It is argued that the most general and effective learning is building a model, reflecting on it, testing it, and sharing with others. Andrea diSessa and Paul Cobb (2004) argued that constructionism is "learning by designing", falling into a category they called "frameworks for action". Richard Noss and James Clayson (2015) suggested six characteristics for such a framework: (1) odelling; (2) accessibility to the modelling process; (3) layering of principles and abstraction; (4) tapping into youth culture; (5) being represented in learner's language; and (6) collaboration. Constructionism has appeared in many other frameworks of activities, for example, the integration of robotics (Camilleri 2017).

Main features. Constructionism is built on the main idea that knowledge is not passively received either through the senses or by way of communication – it is actively built up by the learner during the learning process. A strong feature stressed by Ernst von Glasersfeld (1988) is that the function of cognition is adaptive, serving the subject's organisation of the experiential world, not the discovery of an objective ontological reality.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of technical and analytical competencies shown in Table 1, but the progression of activities is not clarified by this theory. It is not clear that this educational

philosophy would support the development of social and emotional competency activities.

Competencies not covered. The importance of development of social and emotional competencies, and the need for these to be introduced at a young age, is stated more clearly in later definitions and discussions of constructionism. If there is a focus on interaction in learning activities with material resources, at the expense of interactions with peers and others when doing this, then this could hinder the important development of relevant and related social and emotional competencies.

Situated learning

Source. The idea of situated learning was used by John Seeley Brown, Allan Collins and Paul Duguid (1989) to develop a proposal for an instructional model that has implications for classroom practice. Developed further by Jean Lave and Etienne Wenger (1991), situated learning (according to Clancey 1995) is a theory about the nature of human knowledge, dynamically constructed according to experience, often considered within the context of individuals acquiring professional skills.

Main features. According to Jan Herrington and Ron Oliver (1995), the principal theorists and critics of situated learning (focusing on the relationship between learning and the social situation in which it occurs) believe that learning environments that possess certain features elicit effective and usable knowledge: authentic context; authentic activities; expert performances and the modelling of processes; multiple roles and perspectives; collaborative construction of knowledge; coaching and scaffolding at critical times; reflection to enable abstractions to be formed; articulation to enable tacit knowledge to be made explicit; and integrated assessment of learning within the tasks. Situated learning (according to Clancey 1995): (1) should always be integrated with the individual's identity and participation; (2) is constituting an evolving membership and capability to participate in different forms; and (3) is the means of reproduction and development of communities of practice.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of all competencies shown in Table 1, but the theory explored learning across a time-limited development span, rather than a long-term span from 5 years of age.

Competencies not covered. While situated learning would appear to be an ideal background learning theory to frame learning activities to develop all of these competencies, it is difficult to see how this might be applied in practice for young people starting their involvement from, say, 5 years of age. Success would be likely to depend on teachers being competent and confident in terms of creating a resource-rich situated learning environment.

Discovery learning

Source. Discovery learning, based on theories developed by John Dewey (1997), Piaget (1973), and Vygotsky (Rice and Wilson 1999), describes learning as active, process-based, and collaborative. Discovery learning is an inquiry-based, constructivist theory that takes place in problem-solving situations. The learner draws on past experience and existing knowledge to

discover facts and relationships and new truths (Bruner 1961). Models based upon a discovery learning model include: guided discovery; problem-based learning; simulation-based learning; case-based learning; and incidental learning.

Main features. Tracy Bicknell-Holmes and Paul Hoffman (2000) describe five main features of discovery learning: exploring and problem solving; taking responsibility for learning (student driven); and building new from existing knowledge. Its key features are: (1) encouraging active engagement; (2) promoting motivation; (3) promoting autonomy, responsibility, independence; (4) developing creativity and problem solving skills; and (5) tailoring learning experiences.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of all competencies shown in Table 1, but integrating the additional role of instruction is not clear.

Competencies not covered. The approach to discovery learning would need to be carefully considered; expecting discovery to result without appropriate and adequate support would not enable effective development of these competencies. Misunderstandings and misinterpretations could too easily develop and persist without sufficient monitoring and discussion.

Experiential learning

Source. Developed by David Kolb (1984) and Carl Rogers (1969), the model of experiential learning was based on Dewey's concept of "Learning by Doing". Rogers emphasised the importance of experiential learning for knowledge application and not mental learning. He believed experiential learning supported needs of individuals, connected to their change and growth. Kolb described it as a four-stage cyclical process: concrete experience, reflection, abstract conceptualisation, and active experimentation.

Main features. Kolb (1984) proposed six key features of experiential learning: being perceived best as a process, rather than outcomes; an unceasing process based on experience; need for conflict resolution between dialectically contrasting ways of adaptation to the world; an all-inclusive process of adaptation to the world; encompassing connections between the individual and the environment; and involving creation of knowledge as a result of the relationship between social and personal knowledge. While experiential learning is a process of learning through experience, it has been more specifically defined as "learning through reflection on doing" (Patrick 2011: 1003). From the four-stage cyclical model, Kolb's theory combines experience, perception, cognition, and behaviour. Four styles of learning are proposed: (1) assimilators, who learn better when presented with sound logical theories; (2) convergers, who learn better when provided with practical applications of concepts and theories; (3) accommodators, who learn better when provided with "hands-on" experiences; and (4) divergers, who learn better when allowed to observe and collect a wide range of information.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of technical competencies

shown in Table 1, but this theory is based on adult learning perspectives. It is not clear that this educational philosophy would enable the development of learning activities to support analytical or social and emotional competencies.

Competencies not covered. The basis of the learning theory and its focal elements were developed from adult investigations and approaches, rather than from exploration of developmental approaches at younger ages.

Problem-based learning

Source. Wai Leung Tse and Wai Lok Chan (2003) describe problem-based learning (PBL) as an approach where "the problem drives the learning". John Savery and Thomas Duffy (1995) assert that PBL first emerged in the field of medical education in the 1950s. In problem-based learning, teachers facilitate discussion-based learning through questions, asking about and monitoring the problem-solving process (Hmelo 1998; Brilingaitė, Bukauskas, and Juškevičienė 2018). PBL is a constructionist method, allowing students to learn about a subject by exposing them to multiple problems and asking them to construct understanding of the subject through these problems. This kind of learning has been shown to be effective. For example, in mathematics or computing classes where students try to solve problems in many different ways (Hmelo-Silver and Barrows 2006). Many studies have reported adoption of PBL in teaching software engineering courses, especially in Agile software development methods.

Main features. For PBL, activities should be: problem-focused (learning begins by addressing simulations of an authentic, ill-structured problem - built around a problem rather than creating a list of topics); student-centred (the teacher does not command the learning activities, but acts as a facilitator in the whole process); self-directed (students individually and collaboratively assume responsibility for generating learning issues and processes through self- and peer-assessment and access their own experiential knowledge and learning materials); self-reflective (learners monitor their understanding and learn to adjust strategies for learning); and facilitative (instructors are facilitators and not teachers). Defining characteristics of PBL are: (1) learning is driven by challenging, open-ended problems or tasks; (2) problems are context specific; (3) group work is common (learners work as self-directed, active investigators and problem-solvers in small collaborative groups); (4) a key problem is identified and a solution is agreed upon and implemented; (5) teachers adopt the role as facilitators of learning, guiding the learning process and promoting an environment of inquiry.

Match to the competencies of a contemporary software developer. Learning activities could be devised to support the development of all competencies shown in Table 1, but PBL has not been extensively explored for younger age groups.

Competencies not covered. Appropriateness to age would need to be considered. While problem-based approaches are used in effective ways across the age span of compulsory and higher education, in primary education, topic-based and project-based approaches tend to be used more than problem-based approaches.

Connectivism

Source. Connectivism, a digital age theory developed by George Siemens and Stephen Downes, resulted from critiquing perceived limitations of behaviourism, cognitivism and constructivism theories (Siemens 2005; Downes 2012). Connectivism describes how Internet technologies have created new opportunities for learning and sharing information across the World Wide Web and among themselves.

Main features. According to Siemens (2005), key concerns of connectivist learning include: identifying correct and current knowledge; establishing associations across multiple fields, ideas, and concepts; acquitting knowledge through linking varied devices, knowledge bases, or existing networks; and being able to acquire knowledge more vital than already-known knowledge. A key feature of connectivism is that learning taking place online can happen across peer networks; a teacher will guide students to information and answer key questions as needed, to support student learning and sharing. A connected community around shared information can then result. However, how connectivism relates to prior theories has been questioned.

Match to the competencies of a contemporary software developer. It is not clear that learning activities could be devised to support the development of any competencies shown in Table 1.

Competencies not covered. Activities that would depend on uses and understanding of distributed resources and distributed knowledge might not easily support younger age groups.

Mapping Learning Theory Features to Competencies of a Contemporary Software Developer

The mapping exercise in the previous section shows the extent to which a number of background learning theories could be used to frame learning activities in computing (informatics) education. A number of critical concerns are raised from this mapping exercise:

- If Piaget's stages of cognitive development are used to frame learning activity development, as abstract concepts arise at older ages, then computational practices could become decontextualised from a contemporary perspective. In this situation, certain technical competencies will be emphasised earlier than others; how abstract skills are then contextualised later will need to be a key concern.
- Using social constructivist approaches, when and how extents and forms of social intervention and social interaction are introduced would need to be carefully considered, as this will influence outcomes and longer-term approaches to competency development and use.
- For social learning approaches, the importance of developing and subsequently using metacognitive skills will need to be a focus for development from early ages.
- For constructionism approaches, the presence and roles of social interaction will need to be as much a focus as material resource creation, across ages and in all forms of activity (those concerned with technical competencies just as much as those focusing on social and emotional competencies).

- For situated learning, teachers and other supporters will need to be highly competent and confident in creating and using a resource-rich situated context for their learners.
- For discovery learning, independent and unsupported discovery will not be likely to result in effective outcomes, considering the nature of the subject and the misconceptions and misunderstandings that could result.
- Experiential learning, arising from studies of adult interactions, means that it will need to be trialled and tested extensively across age ranges.
- Problem-based learning has been used across educational practice widely (although limited at primary age level, and in some countries in the secondary age level).
- Connectivism is likely to be difficult to apply for younger age ranges as this theory is concerned with how individuals manage distributed (online) resources and knowledge across networks.

These concerns indicate that any underpinning learning theory should embody social interactivity and development, creative practice, and constructionist principles. This might be termed *social creative constructionism*, an emerging theory for further future research.

Conclusions

It is not possible to say that competencies could not be developed through educational practices framed by one or another learning theory. However, background learning theories used to frame developments of learning activities can have a major effect, easily influencing (positively or negatively) forms of interaction and, hence, either enhancing or limiting developments of certain competencies. There are some limitations that could arise when planning a progression of activities if cognitive constructivism is used as the only underpinning learning theory (Table 2). There are also limitations arising from using other learning theories: for technical competencies, progression of learning activities is not clearly identified (although depth of quality is identified in some cases for some age groups); for analytical competencies, support for creation of activities is not clear; and for social and emotional competencies, creation of activities across the age span is not clear.

Based on this review and analysis, whilst a completely new learning theory might not be shown as an absolute necessity at this time, it is clear that our educational context has shifted. So, any background framing educational theory needs to adequately and appropriately consider particularly how analytical, social and emotional competencies will be developed across age ranges. How competencies not easy to develop at young ages will be appropriately contextualised as early as possible will also require careful consideration. Future research will clearly need to consider how length of learning is conceived across the age range, as well as depth of learning, meeting the demands of *social creative constructionism*.

REFERENCES

Ahmed, F., Capretz, L. F., Bouktif, S., and Campbell, P., 2013. Soft Skills and Software Development: A Reflection from Software Industry. *International Journal of Information Processing and Management* 4(3): 171-191.

Bandura, A., 1977. *Social Learning Theory*. Englewood Cliffs, NJ: Prentice Hall.

Berger, P., and Luckmann, T., 1991. *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. London: Penguin Books.

Bicknell-Holmes, T., and Hoffman, P. S., 2000. Elicit, Engage, Experience, Explore: Discovery Learning in Library Instruction. *Reference Services Review* 28(4): 313-322.

Bodrova, E., and Leong, D. J., 2012. Tools of the Mind: Vygotskian Approach to Early Childhood Education. In J.L. Rooparine and J. Jones (eds.), *Approaches to Early Childhood Education* (6th ed.), Upper Saddle River, NJ: Merrill, 2012, pp. 241-260.

Brilingaitė, A., Bukauskas, L., and Juškevičienė, A., 2018. Competency Assessment in Problem-Based Learning Projects of Information Technologies Students. *Informatics in Education* 17(1): 21-44.

Brown J. S., Collins, J., and Duguid, P., 1989. Situated Cognition and the Culture of Learning. *Educational Researcher* 18(1): 32-42.

Bruner, J. S., 1961. The Act of Discovery. *Harvard Educational Review* 31: 21-32.

Camilleri, P., 2017. Minding the Gap: Proposing a Teacher Learning-Training Framework for the Integration of Robotics in Primary Schools. *Informatics in Education* 16(2):165-179.

Clancey, W. J., 1995. A Tutorial on Situated Learning. In J. Self (ed.), *Proceedings of the International Conference on Computers and Education*. Charlottesville, VA: AACE, pp. 49-70.

Department for Education, 2013. National Curriculum in England: Computing Programmes of Study. Available at https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

Dewey, J., 1997. *Democracy and Education*. New York, NY: Simon and Schuster.

diSessa, A. A., and Cobb, P., 2004. Ontological Innovation and the Role of Theory in Design Experiments. *Journal of the Learning Sciences* 13(1): 77-103.

Downes, S., 2007. What connectivism is, Half An Hour, February 3, Retrieved from http://halfanhour.blogspot.ro/2007/02/what-connectivism-is.html

Downes, S., 2012. *Connectivism and Connective Knowledge*. Available at https://www.downes.ca/files/books/Connective_Knowledge-19May2012.pdf

Gauvain, M., 2008. Vygotsky's Sociocultural Theory. In M. M. Haith and J. B. Benson (eds.), *Encyclopedia of Infant and Early Childhood Development*, Vol. 3, Oxford, UK: Elsevier, 2008, pp. 404-413.

Glasersfeld, E. von, 1988. The Reluctance to Change a Way of Thinking. *Irish Journal of Psychology* 9(1): 83-90.

Gredler, M., 2008. Vygotsky's Cultural-Historical Theory of Development. In N. J. Salkind (ed.), *Encyclopedia of Educational Psychology*, Vol. 1, Thousand Oaks, CA: Sage Publications, pp. 1011-1014.

Herrington, J., and Oliver, R., 1995. Critical Characteristics of Situated Learning: Implications for the Instructional Design of Multimedia. In *ASCILITE 1995 Conference*, December 3-7, 1995, University of Melbourne, Melbourne, pp. 253-262.

Hmelo, C. E., 1998. Problem-Based Learning: Effects on the Early Acquisition of Cognitive Skill in Medicine. *Journal of the Learning Sciences* 7: 173-208.

Hmelo-Silver, C. E., and Barrows, H. S., 2006. Goals and Strategies of a Problem-Based Learning Facilitator. *Interdisciplinary Journal of Problem-based Learning* 1: 21-39.

Kalpana, T., 2014. A Constructivist Perspective on Teaching and Learning: A Conceptual Framework. *International Research Journal of Social Sciences* 3(1): 27-29.

Kolb, D. A., 1984. *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall.

Lamb, S., Maire, Q., and Doecke, E., 2017. *Key Skills for the 21st Century: An Evidence-based Review*. Melbourne, VIC: Victoria University, Australia.

Lave, J., and Wenger, E., 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.

Martinez, S. L., and Stager, G., 2013. *Invent to Learn: Making, Tinkering, and Engineering in the Classroom*. Torrance, CA: Constructing Modern Knowledge Press.

Noss, R., and Clayson, J., 2015. Reconstructing Constructionism. *Constructivist Foundations* 10(3): 285-288.

Papert, S., 1986. *Constructionism: A New Opportunity for Elementary Science Education*. Massachusetts Institute of Technology, Media Laboratory, Epistemology and Learning Group.

Patrick, F., 2011. *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches.* Hershey, PA: Information Science Reference.

Piaget, J., 1936. *La Naissance de l'Intelligence chez l'Enfant.* Neuchatel, Paris: Delachaux & Niestlé.

Piaget, J., 1952. *The Origins of Intelligence in Children*. Transl. by M. Cook. New York: International University Press.

Piaget, J., 1973. *To Understand is to Invent: The Future of Education*. Transl. by G.-A. Roberts. New York, NY: Grossman.

Rice, M. L., and Wilson, E. K., 1999. How Technology Aids Constructivism in the Social Studies Classroom. *Social Studies* 90(1): 28-33.

Rogers, C. R., 1969. *Freedom to Learn: A View of What Education Might Become*. Columbus, OH: Charles E. Merrill.

Savery, J. R., and Duffy, T. M., 1995. Problem Based Learning: An Instructional Model and Its Constructivist Framework. *Educational Technology* 35(5): 31-38.

Siemens, G., 2005. Connectivism: A Learning Theory for the Digital Age. *International Journal of Instructional Technology and Distance Learning* 2(1): 3-10.

SkillScan, 2012. *Chart of Skill Categories, Skill Sets and Sample Career Options*. Available at: https://www.skillscan.com/sites/default/files/chart-of-skill-sets.pdf

Surakka, S., and Malmi, L., 2004. Cognitive Skills of Experienced Software Developer: Delphi Study. In A. Korhonen and L. Malmi (eds.), *Kolin Kolistelut - Koli Calling 2004*: *Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*, Helsinki: Helsinki University of Technology, pp. 37-46.

Tse, W. L., and Chan, W. L., 2003. Application of Problem-Based Learning in an Engineering Course. *International Journal of Engineering Education* 19(5): 747-753.

Vygotsky, L. S., 1978. *Mind in Society: The Development of Higher Psychological Processes*. Translated by M. Cole, V. John-Steiner, S. Scribner, and E. Souberman. Cambridge, MA: Harvard University Press.

Wing, J. M., 2008. Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366: 3717-3725.

INFORMATIKOS PRAKTIKOS, UGDYMO TEORIJOS IR MOKYMOSI PAŽANGA

Don Passey, Valentina Dagienė, Loice Victorine Atieno, Wilfried Baumann

Santrauka. Daugelis šalių imasi mokyti informatikos (arba kompiuterių mokslo) mokyklose pradėdamos nuo pradinių klasių ar net darželinio amžiaus. Tačiau nėra pakankamai aišku, kokiomis ugdymo filosofijos prielaidomis grindžiamos atitinkamos mokymų programos, nėra aiškiai įvardyta, kokiomis mokymo teorijomis remiamasi bendrųjų ugdymo programų dokumentuose. Dauguma bendrųjų ugdymo programų Vakarų šalyse yra grindžiamos kognityvinio konstruktyvizmo principais parenkant veiklas pagal amžiaus tarpsnius, atsižvelgiant į sensomotorinio, priešoperacinio, konkrečių operacijų ir formalių operacijų mąstymo stadijas. Šią ugdymo sistemą papildo socialinis konstruktyvizmas ir konstrukcionizmas, kurie fundamentaliai prisidėjo plėtojant praktines informatikos ugdymo veiklas. Straipsnyje apžvelgiamos atrinktos ugdymo teorijos ir tyrinėjamos savybės, kurios turėtų atsirasti informatikos ugdymo programose, siekiant ugdyti praktinius programavimo ir programinės įrangos kūrimo įgūdžius.

Pagrindiniai žodžiai: informatikos praktika, ugdymo filosofija, mokymo(si) teorijos, mokymosi pažanga, programinės įrangos kūrėjo kompetencijos

Įteikta 2018 08 20
Priimta 2018 12 03