Keeping software engineering students in touch with not only *what* they are to learn, but with *why*

Syed Waqar Nabi
School of Computing Science
University of Glasgow
Glasgow, UK
syed.nabi@glasgow.ac.uk

Joseph Maguire
School of Computing Science
University of Glasgow
Glasgow, UK
joseph.maguire@glasgow.ac.uk

Steve Draper
School of Psychology
University of Glasgow
Glasgow, UK
steve.draper@glasgow.ac.uk

Quintin Cutts
School of Computing Science
University of Glasgow
Glasgow, UK
quintin.cutts@glasgow.ac.uk

***Abstract*—We needed to design a module covering both discrete maths and algorithms for a Work Based Learning (WBL) Software Engineering degree programme, where students spend half their time at their employers' workplace. This entails deciding the order of topics, and explaining that to students. Sequence in the course is only one kind of relationship between topics in the module; relevance to work is another. Such explaining is a neglected educational issue. This paper addresses how to convey these relationships to learners. Because of the WBL context, it is inevitable that questions about how pieces of academic learning relate to the work context (which these students are already experiencing) arise immediately and vividly for them. This is important because it affects learner motivation, sometimes greatly, and also whether they are likely to learn in a surface or a deep mode. A given learner's grasp of these relationships is both individual and changes over time. Because of this, the general pedagogical approach discussed here is "little and often": raising it a number of times, with an emphasis not on one exhaustive and rigid view but on how different students often see it differently, and encouraging each to gradually develop their own view. The concept map diagram in this paper illustrates only one example of how the web of connections might be represented and conveyed during the course.**

***Index Terms*—work-based learning, software engineering education, professional learning**

I. Introduction

A perennial complaint about software engineering education from employers is that graduates from such programmes often lack the perceived skills required by industry [1]. The increasing significance of software engineering to advanced economies around the world has resulted in the governors of such economies to promote industry-linked degree programmes. These programmes typically expect employers and educators to partner in the delivery and development of software engineering graduates. Such industry-linked software engineering programmes are still delivered alongside traditional academic programmes. This provides an opportunity for students to elect either a more applied or theoretical route in their journey to become a software engineer.

The expectation is that individuals who opt for the industry-linked or more applied software engineering education route, expect just that, the argument being that such students will have different motivations and expectations than those who opt for a more academic, theoretical route. Students enrolled on an applied programme are more likely to be motivated to excel in their employed role, less concerned with theory and more focused on learning practical abilities that allow them to demonstrate their competency to their superiors.

The concern is that students on such industry-linked programmes may become demotivated or disengaged from a course or programme, if they perceive the content they encounter as not aligned with their professional role. They may perceive some concepts, such as number theory and graphs, as pointless and irrelevant, potentially fueling frustration and disengagement. A situation that will ultimately result in a student not fully grasping critical theoretical concepts.

These are the present concerns of the authors as they are faced with the challenge of developing a course that encompasses topics, including discrete mathematics and algorithms, for students enrolled on an industry-linked programme. The perceived solution is to engage students in an on-going process of considering not only the content they are learning, but *why* they are learning it. The expectation is that this will support learner motivations and ensure on-going engagement with the course. Consequently, the contributions of this paper are:

- consideration of the literature to do with learner motivation in line with software engineering;
- a proposed approach for the design of courses that act to enlist learner motivation;
- discussion around the notion that learner motivation is a neglected educational issue relevant beyond software engineering education.

The expectation is that by sharing this early work with others we can engage in a conversation as to the optimal approach to deliver highly theoretical content without negatively impacting on learner motivations when delivering highly applied and industry-linked degree programmes.

II. Background

The importance of learner motivation toward engagement with course content and grasping of theoretical concepts is unlikely to surprise many academics or instructors. However, despite its relevance and importance it often goes under-appreciated when considering how students engage with course content. The significance to software engineering education is also pertinent as some students may perceive theory

as not relevant to their profession and so pay less attention to it, despite such theory being valued by their employers [2].

Nevertheless, learner motivation is important in terms of the effort they will apply to engage and learn. The challenge is that students may not share all the same motivations. Hars and Ou explored the motivations of those individuals that participated in large-scale open-source programming projects [3]. The majority of participants in their survey contributed to open-source programming projects because it was part of their employment while others participated to learn more, while many more simply participated to pass the time.

Hars and Ou probed further into the reasons why individuals contribute to open-source projects and found expected motivations, such as to improve programming and code reading ability, as well as more unexpected motivations, such as to improve English language skills or to improve team collaboration skills. The expectation is that if students entered into open-source projects with such motivations in mind, if they feel the experience is not delivering on such motivations they are likely to disengage.

Disengagement from large-scale open source programming projects is another area of on-going investigation for researchers, specifically why do people abandon such project when they enter them with enthusiasm. Steinmacher et al. explored the reasons why individual disengage or abandon large-scale open source programming projects [4]. The authors found that the most significant barriers to on-going contribution was not so much motivation, but having no sense of where to begin or how to contribute to a project.

Ye and Kishida argue open-source projects should be structured to ensure as many independent tasks as possible, rated in terms of difficulty to support learners progressing through them [5]. They argue that learning is a central and crucial motivation for participating in open-source projects. Furthermore, the authors argue such projects offer students access to rich communities of practice, affording the opportunity of "Legitimate Peripheral Participation" in the development of large scale software projects. Ye and Kishida articulate that educators should emphasise that the *real* opportunity of such experiences is to observe the work of experienced software engineers as well as how to communicate effectively with them, rather than simply to focus on technical skills.

Consequently, in order to sustain learner motivation for such projects they need to be structured to support development, but more importantly emphasis should be on opportunities that would be valuable for learners. The concern is that educators may emphasise such opportunities, say through course design and assessments, but inadvertently turn students off if it is not aligned with their expectations and motivations. If students perceive the most important aspect of being a good software engineer is technical competency, then educators focusing on other aspects may only cause students to disengage.

Li, Ko and Jiamin argue that most learners are motivated to become great software engineers, without any significant appreciation of what it is to be a great software engineer [6]. The authors interviewed 59 experienced engineers across 13 different divisions in Microsoft. The survey revealed many attributes that students may expect, such individuals should drive to create elegant code, see the big picture and be creative. However, Li, Ko and Jiamin also indicate that there were many aspects that students may not appreciate such as significant social aspects, specially knowing how to set the expectations of superiors, managing subordinates as well as knowing how to collaborate and communicate, e.g. who to ask for help.

The expectation is that a student may perceive asking for help as demonstrating incompetence, but employers would prefer individuals to collaborate and seek help from others rather than spending hours searching for a solution. Begel and Simon explored the transition software engineering students made from university to industry [7]. They asked eight junior software engineer to log their activities in their new role. Begel and Simon reported that students spend a great deal of their day on communication, documentation and working on bugs.

In interviews with the participants, interesting aspects arose - specifically some junior software engineers argued that breadth was more important than depth. That 'deep diving' on any one topic was expensive and could potentially slow progress. Furthermore, some of the junior developers argued that knowing when to seek help and from whom was another important aspect, but junior engineers they do not want to appear incompetent. The interesting aspect is that if students were to encounter such opportunities during their education, they may lose motivation from them.

However, this is not necessarily the same for all learners on the same course or programme. Dziallas and Fincher demonstrate the different perspectives of what two different individuals felt they had gain from their computing science degree after graduation [8]. In particular, one former student, Henry, felt that they initially had "an over-riding sense of disappointment" in terms of what they had experienced, only to report a decade later that while they could not see the relevance of content at the point of exposure, they could now due to experience.

This suggests that software engineering students, especially those engaged in applied programmes, may be better served by nurturing their understanding not only of the content they are engaged with, but of why they are engaged with it. Given the differing perspectives and motivations of learners, it seems necessary for students to form their own connections between content and concepts being presented and their own motivations. Moreover, such consideration should not happen at the start or end of a course, but frequently throughout it.

III. Identifying The Challenges

The key challenge we address is this: how do we achieve *meaningful*, *motivated*, and *contextualized* learning in the context of a Work-Based Learning (WBL) degree program. From the teacher's perspective, the challenges may be discussed along the following dimensions:

Work-Based Learning versus Higher Education: The teacher is faced with a tension, whether real or perceived, between satisfying two requirements that, while not always at

complete odds, will not always conveniently line up either. One one side, there is the expectation shared by the WBL students and their employers that the module (and the overall degree program in general) would deliver skills and knowledge that complement the workplace learning. On the other hand, there is an expectation that higher education institutions are responsible for creating a broad and sound knowledge foundation based at least in part on theoretical knowledge, and will incorporate academic rigour in the process of doing so. Designing a module structure that satisfies these expectations is challenging, and it is quite possible that, in the absence of a deliberate and well planned attempt to balance these requirements, a university teacher may default to the more familiar, academically rigorous approach which could end up being disconnected from workplace environment that the student is concurrently experiencing.

Universal Truths versus Applied Skills: Many academics, especially those drawing from a personal research focus on a theoretically grounded subject, would naturally emphasize what may be called the *Universal Truths* of their discipline. These are axioms, theories, laws, and concepts that are intrinsically motivated, have a long shelf life, and are not necessarily driven by external, industry-driven expectations. Such an emphasis naturally integrates well with the overall environment of academic rigour in a higher education institution. Here too there is a tension that resonates with that between WBL and higher education discussed earlier. The WBL student (and their employers) may be presumed to be more interested in acquiring skills that are applicable to their work as they are experiencing it then, or can foresee in the near future, which is fair. The teacher is thus faced with the challenge of reconciling what is a somewhat idealistic, and—from their perspective—an internally motivated emphasis (tilted towards *universal truths*) with an externally motivated requirement of imparting applied skills driven by a utilitarian vantage point.

The challenges of WBL in higher-education manifest somewhat differently for the students:

The Temporal Aspect: WBL students, especially in the early years of a degree program, may have a disproportionate focus on the immediate relevance of what they are learning at the university. While there are some modules designed to address this focus, a higher education degree would naturally cover topics that may become relevant much later, possibly many years after they have actually been studied. There is evidence suggesting that students can lose motivation if this longer-term view is not made visible to them (e.g. [8]), and this problem is accentuated in a WBL context where the students are meant to, and expect to, apply their knowledge to the work place *while* they are pursuing their higher education.

Personal Viewpoints: Universities in general presume a shared and more or less uniform learning experience for students who are part of the same cohort. It can be said however that such a presumption in most cases is not fair, and almost certainly not when it comes to WBL students, as every student is on a unique and personal learning journey. WBL students are working at different organizations and job roles *while* they are studying at University. In addition to everyone being different in terms of background knowledge and cognitive abilities, they will also have a unique and personal experience of their workplaces. This situation can be expected to lead to a wider-than-typical variation in terms of perspectives, expectations and motivations in the classroom.

The challenges laid out above are not independent, and there may be other dimensions to the overall issue. We propose that framing the challenges in the form outlined here however is a useful contribution, as it contextualizes the discussion on addressing the larger challenge of WBL in higher education, and informs how the effectiveness of the various approaches may be evaluated. Having laid out the challenges of WBL in higher education from a teacher and from a student perspective, we will now propose one particular method that we expect to address them at least in part.

IV. Sample Activities Using Concept Maps

The specific module which forms the backdrop of this work is called *Practical Algorithms*. It brings together topics that are conventionally taught in two modules, *Discrete Mathematics* and *Algorithms and Data Structures*, into a single one. The target audience of this module are students in a Software Engineering WBL setting at the University of Glasgow.

We propose the use of *concept maps* as a suitable teaching and learning tool. A concept map is simply a graphical tool that is useful in illustrating the relationships between *concepts*. It was developed by Novak in the 1970s as an aid to understanding and following changes in childrens' understanding of science [9]. It is based on the cognitive development theory of *subsumption* proposed by Ausubel, where the key idea is that learning takes place by *assimilation* of new concepts into an existing framework and concepts. Concept maps have been shown to facilitate *meaningful learning*, as opposed to what may be called *rote learning*. Using concept maps, Novak noted a surprisingly positive impact on the ability to *utilize* knowledge in new contexts, and also the ability to *retain* knowledge over long periods of time [10].

Specifically for WBL in higher education, we propose that concept maps can be used by the teacher to:

- Develop a personal, internal model of the topics in a module and their inter-connectedness, as well as their connections to other modules of the program and applied topics relevant to the workplace industry.
- Develop an over-arching narrative for the module such that it meets the requirements of both the WBL and the higher education context.
- Represent connections from universal truths based on theoretical foundations to applied knowledge and skills required at the workplace.
- Identify a suitable order of delivering topics in a module.

It can also be used by the student, to:

- Appreciate the temporal aspect of knowledge acquisition, by allowing them to clearly see concepts from the module connected to, and hence expected to be useful in, areas of knowledge that may be relevant in the future.

- See connections between the concepts being taught to their own personal and unique job roles at the workplace.
- As the module progresses, track and evaluate their own understanding of the relevant concepts and their connections to their job roles at work.
- Have a deeper, more *meaningful* learning experience, leading to a better *utilization* and *retention* of knowledge.

The concept map shown in Figure 1 represents one example of how a web of connections might be conveyed during the module in service of these goals. The concept map is roughly divided into three colour-coded clusters, representing the partitioning of the module into three highest level *concepts* (or equivalently, *topics*): discrete mathematics (top, green), data structures (magenta, left), and algorithms (blue, right). The grey boxes represent topics students would encounter in other modules in the program, and may already be interacting with at their work. Orange lines are used to highlight connections that cross the top-level subject domains, so we can see e.g. that *algorithm analysis techniques* from the *algorithms* domain is connected to *probability* and *counting*, topics from the *discrete mathematics* domain. They grey dotted lines show connections to concepts outside the domain of this module.

Appropriately capturing the *nature* of the connection between various concept in a concept map is crucial in ensuring its effectiveness. There are a number of such relationships that are at play right from the beginning of the module:

- A is a part of B[1]
- A uses B
- A is based on B
- A is a type of B
- A informs B

A concept map like the one presented can be used in a number of ways. An opening exercise might be to present the diagram and get each learner individually to express e.g. which concepts they think useful; how is each concept useful, either at present in their workplace, or potentially in the future; and which they are scared of approaching.

The answer to these questions will help the student in making personalized connections between concepts in the module and their own work, and also identify topics they might want to focus on and possibly seek help with from the instructor or from peers in a group activity. The instructor can also find the feedback from such questions useful in identifying focus areas in the module.

Another similar exercise would be to ask learners individually to write down the job roles which they can identify in their own workplace; and then to try to link them to parts of the module. We have put place-holders in the concept map in Figure 1 to indicate how this activity might be carried out.

In line with our approach of "little and often", the concept map could be revisited every time a topic transition is made in class, and students could be queried about how (or if) their view of the concept map has changed. The students may be handed out incomplete maps with only the topics listed but not the connections, and they could be asked to make the connections individually or in groups.

We also foresee using the concept maps in a hierarchical fashion, by creating topic-level concept maps. At this level, an individual concept map will represent one specific topic, typically covered in 1–2 lectures. This concept map will be developed *by the students*, possibly using templates where sub-topics have been placed but not connected. It might be done throughout the module (and the year, and the 4-year programme), encouraging each student to gradually develop their own view.

V. Discussion

The use of concept maps in the manner proposed in this work is consistent with some core educational theories, which we briefly discuss here:

Constructivism is used in various overlapping senses in the education literature, but the core issues are: We cannot plug in new knowledge to a human learner, as we can plug in a database to a laptop via a USB port. Only the learner can do the insertion. The reason for this is that the hard part of learning isn't absorbing new information but discovering *where* and *how* in the learner's existing knowledge it should be connected. Hence effective teaching must consider *both* the destination (the knowledge to be learned) *and* each learner's starting point: their prior knowledge.

Connectivism refers to various related ideas including viewing knowledge as all about a network of partial ideas; that today's "connected" world is significantly different as a learning context; and using such human connectedness is more important than ever so that each learner compares and contrasts their grasp of new knowledge with that of other learners. This peer process not only exercises the new knowledge from the start, and quickly corrects many incipient misconceptions, but also allows the learner to be aware of where the knowledge means something different to others.

Deep and surface knowledge; the metaphor implies that deep equals good, surface equals bad. But it is important to realise that surface learning is valuable, even though having both deep and surface is better. If all you know is a term (e.g. "connectivism") without understanding its meaning, you can look it up at any time including in the distant future. Without the term you don't know there is something to learn, and you can't find out about it. The definition of "deep learning" used here is the number of *types* of link a new piece of knowledge is connected to in the learner's mind: understanding in more detail (which is "deep" in a sense common in computing) is only one type of link, and not always the most important.

Concept maps are an intuitive tool that are effective as they draw on well established learning theories. We have discussed our attempt to do so here in a very specific context of a somewhat abstract module in a WBL scenario. However, it is important to keep the perspective that other tools and methodologies, or a certain combination of them, may be more

[1]In our diagram, we have used colour coding the denote this relationship as it reduces the clutter. So although there are no explicit connections indicated, all the green nodes *are a part of* the larger concept "Discrete Mathematics".

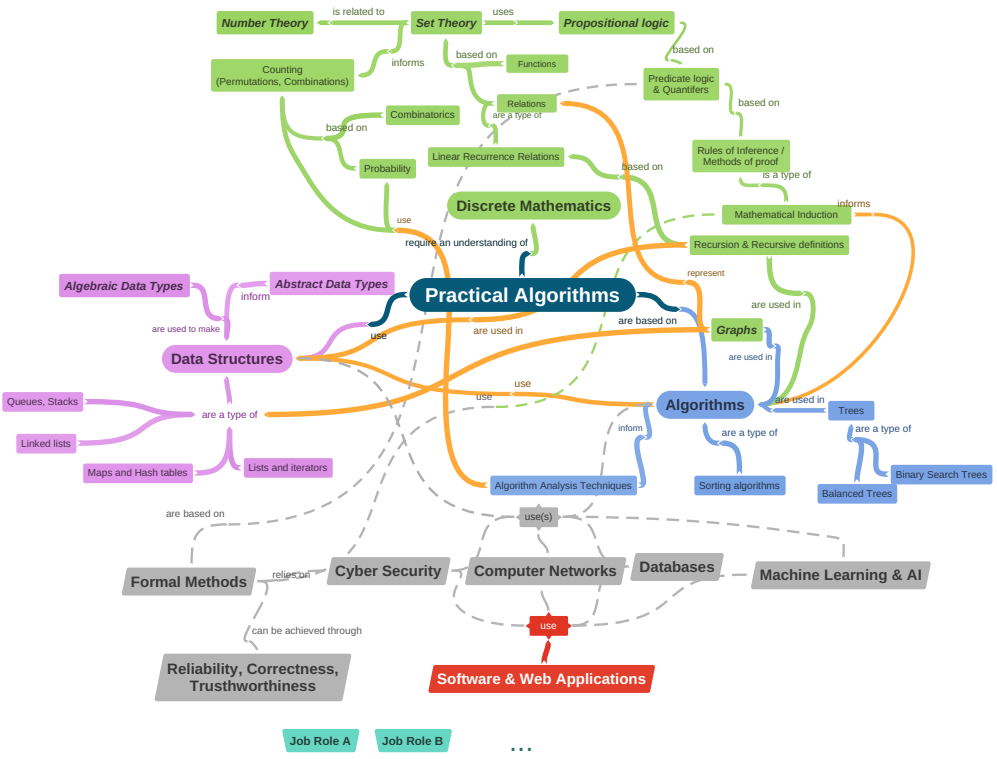

Fig. 1. Module-level concept map for the *Practical Algorithms* module. The three main topics are color-coded (magenta for data structures, green for discrete maths, and blue for algorithms), and the directional connections indicate the nature of the relationship. Orange lined denote connections that cut across the top-level topics, and grey boxes and connections show links to topics outside the domain of this specific course.

suitable for a certain module and program. The primary goal is ensuring that pedagogical methods have been informed by well-founded educational theories.

VI. Conclusion

We have presented our contributions to pedagogy in Software Engineering set in, with a focus on *meaningful* and *motivated* learning. Our specific context is a theory oriented module in a work-based learning setting. We gave our view on what we consider are the key challenges in such a scenario, both from the teacher and the learner perspective. Among others, one of the key issues is ensuring that the learner is able to make connections between concepts they already know, those that they are learning in the module, and, very importantly, to the skills required at their workplace. Informed by well recognized educational theories of *Constructivism* and *Connectivism*, we proposed the use of *concept maps*, and identified how they could be used both by the teacher and the learner. It is our expectation that such a methodology will be instrumental, likely in combination with other pedagogical approaches, in achieving the goal of properly motivated, contextualized, deep and meaningful learning. Our work is still in its early stages, we will continue the investigation by carrying out empirical studies, and will also look at other tools and methodologies that could complement our proposed approach.

References

[1] A. Radermacher and G. Walia, "Gaps between industry expectations and the abilities of graduates," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013, pp. 525–530.

[2] M. Barr and J. Parkinson, "Developing a work-based software engineering degree in collaboration with industry," in *Proceedings of the 1st UK & Ireland Computing Education Research Conference*, 2019, pp. 1–7.

[3] S. O. Alexander Hars, "Working for free? motivations for participating in open-source projects," *International Journal of Electronic Commerce*, vol. 6, no. 3, pp. 25–39, 2002.

[4] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles, "The hard life of open source software project newcomers," in *Proceedings of the 7th international workshop on cooperative and human aspects of software engineering*, 2014, pp. 72–78.

[5] Y. Ye and K. Kishida, "Toward an understanding of the motivation of open source software developers," in *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE, 2003, pp. 419–429.

[6] P. L. Li, A. J. Ko, and J. Zhu, "What makes a great software engineer?" in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 700–710.

[7] A. Begel and B. Simon, "Novice software developers, all over again," in *Proceedings of the fourth international workshop on computing education research*, 2008, pp. 3–14.

[8] S. Dziallas and S. Fincher, "Accountable disciplinary knowledge in computing education: A case-comparative approach," in *Proceedings of the 2019 ACM Conference on International Computing Education Research*, 2019, pp. 1–9.

[9] J. D. Novak, "Concept mapping: A useful tool for science education," *Journal of Research in Science Teaching*, vol. 27, no. 10, pp. 937–949, 1990. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/tea.3660271003

[10] J. D. Novak and A. J. Cañas, "Theoretical origins of concept maps, how to construct them, and uses in education," *Reflecting Education*, vol. 3, no. 1, pp. 29–42, 2007.