TEACHING & LEARNING RESEARCH INITIATIVE

NĀU I WHATU TE KĀKAHU, HE TĀNIKO TAKU

Copy, cut, and paste: How does this shape what we know?

Elaine Khoo, Craig Hight, Rob Torrens, and Bronwen Cowie

August 2015

Introduction

Copy, cut, and paste are functions naturalised and embedded across different software applications but are poorly understood as tools that shape our engagement with knowledge, culture and society in the 21st century (Livingstone, Wijnen, Papaioannou, Costa, & Grandio, 2014). Most people develop proficiency with ubiquitous software packages, such as those on cell phones, informally through their everyday engagement. Tertiary students as so-called "digital natives" (Prensky, 2001) are assumed to be able to translate this informally developed knowledge and skills into formal settings to successfully accomplish learning tasks. Educators often assume that students already possess the necessary skills and conceptual frameworks to learn with and through generic software packages, and tend to neglect how the affordances of different software shape the ways students "perform" the software (Adams, 2006). Emerging evidence internationally and locally indicates a dearth in this digital generation's basic academic literacy skills for successful learning despite their technological competency (Kvavik, 2005). Given the political, technological, financial, staff workload, and student learning implications that come with the adoption of ICTs in the education sector, it is imperative for universities to understand how to close the gap for students and ensure that technology is equitably and effectively used to support learning (Coolbear, 2008).

We propose the following model for software literacy as the repertoires of skills and understandings needed for students to be critical and creative users of software packages and systems in a software saturated culture. We hypothesise that there are three progressive tiers of development towards software literacy: (1) a basic skill level where a learner can use a particular software; (2) an ability to independently troubleshoot and problem-solve issues faced when using the software; and finally, (3) the ability to critique the software, including being able to apply such critique to a range of software designed for a similar purpose and to use these understandings for new software learning. The third tier involves the ability to identify affordances and their implications (including the constraints) of particular software and identify ways to both apply and extend its use such that it is relevant and meaningful to a wider range of learning purposes, tasks and contexts.

This conceptual model is a response to current limitations in digital literacy frameworks which do not identify the implications of software. Students may not be aware of the full implications of the affordances and constraints offered through particular software. No studies to date that we know of raise the role of student understanding of how software and its affordances influence knowledge representation, generation, and critique. Many studies have been conducted on information literacy and on ways of mastering software (Underwood, 2009), but the role of software itself tends to be taken for granted and is not questioned. The study of software is only now emerging as a field of study (Fuller, 2008; Manovich, 2008).

We know very little about how students develop the skills and expertise needed to attend to the features and use of software (as application, platform, and architecture) to complete everyday tasks. There is evidence that the ubiquity of software and ICT tools has led students to adopt a range of informal approaches to meet their learning needs (Peeters et al., 2014). Research also indicates that students' formal software learning backgrounds are diverse (Khoo, Johnson, Torrens, & Fulton, 2011). Student knowledge and use of software and technology is highly specific to their formal and informal educational, social, and cultural contexts for learning and use (Jones, Ramanau, Cross, & Healing, 2010; Valtonen, Dillon, Hacklin, & Väisänen, 2010). The challenge is thus posed for educators to adopt pedagogical strategies that build on this diversity. This research is therefore important in its investigation of how students develop the knowledge and skills to use software and the extent to which they are able to apply and extend these to successfully learn and act in formal tertiary learning contexts.

Research design

This research aimed to explore, examine, and theorise on how the notion of software literacy is understood, developed, and applied in tertiary teaching and learning contexts, and the extent to which this understanding is useful when translated into new contexts of learning with and through software. The research aim was translated into the following questions:

1. To what extent, and how, does student software literacy develop and impact on the teaching and learning of discipline-specific software in formal tertiary teaching settings?

   The subsidiary underpinning research questions were:

   a. Are lecturers aware of the implications of the affordances of the software they are using, specifically, in regards to using PowerPoint?

   b. Are students aware of the implications of the affordances of the software they are using, specifically, in regards to using PowerPoint?

3. How and in what ways do lecturers model attention to and use of different aspects of software affordance in a course which utilises discipline-specific software?
4. What software literacy do students consider they learnt as part of the case study tertiary course(s)?

Questions 2 and 3 focus on the impact of discipline-specific software on teaching and learning, and how students subsequently come to conceptualise and experience their field of study.

There were two phases in this exploratory study designed to address these questions.

Phase 1 addressed the first research question in terms of exploring and understanding the ways lecturers and students become aware of and develop software literacy understandings and skills about PowerPoint. This was selected as a commonly used application that provided an inclusive context for discussing the role of software in affording and constraining student opportunities to learn.

Phase 2 examined how discipline-specific software literacy develops in a formal learning environment and the extent this development fitted with our hypothesised model. Phase 2 addressed research questions 2 and 3. The intention was to unpack if and how students develop and use discipline-specific software literacy, understand the influence of software on the way they make sense of disciplinary knowledge, and whether their learning trajectories fit with the hypothesised tiers of software literacy.

Data collection

The study used a qualitative interpretive methodology to frame the collection and analysis of data because this method allowed for careful attention to lecturer and student perspectives (Maykut & Morehouse, 1994). Further, a case study approach was adopted to allow the research team to develop an in-depth understanding of participants' lived experiences and transformations throughout the period of the study (Gall, Borg, & Gall, 1996).

Case studies were developed for two different disciplines (engineering and media studies) based on collaboration with lecturers who were keen to examine the notion of software literacy. Both cases began with a focus on PowerPoint and moved to focus on the teaching and learning of commonplace discipline-specific software—Final Cut Pro and Adobe Creative Suite (media editing applications) in media studies, and Solidworks (a computer-aided design, or CAD, software) in engineering. Both programmes are characterised by high enrolments of students with diverse backgrounds at entry level (180 and 104 students respectively in 2013) but differ in terms of professional pathways. Both disciplines make extensive use of PowerPoint in lectures, and both provide teaching about the specific applications. Both disciplines use laboratory-based formal training, and provide resources for additional informal training.

Both applications demand, and support, visual and spatial thinking but in different ways and for different purpose. However, these applications are positioned very differently in relation to disciplinary learning and knowledge. For example, SolidWorks is compulsory for engineering undergraduates while media editing software is an elective within media studies. The learning of the applications provided differing contexts to examine the notion of software literacy as it relates to the use of software tools to create and manipulate images and diagrams

Multiple data were collected to address the research questions through:

- lecturer interviews and a peer reflexive workshop
- observations of lectures and laboratory sessions
- online student surveys
- student focus groups
- student produced work
- ongoing informal interviews with lecturers and students.

As part of enhancing the quality and interpretation of the data collection and interpretation, strategies such as triangulation across multiple data sources and researchers were employed alongside documenting an audit trail, regular team meetings and peer debriefing (a peer reflexive workshop), and member checking of data by lecturer participants to verify their interview transcripts (Lincoln & Guba, 1985).

Analysis

A sociocultural theoretical framework provided the overarching analytical frame to guide the data collection and analysis (Wertsch, 1998). A sociocultural view of learning as a mediated action (Wertsch, 1991a) was pertinent to this study where the focus was on how participants learned and used software as a means for accomplishing a range of goals (Wertsch, 1991b, 1998). This allowed us to understand the functioning of the individual in relation to his or her unique sociocultural setting and how the setting in turn influences and transforms the individual in significant ways. In our study, the focus is thus on how people-in-action are using software. We drew on Miettenen (2001) to view software as an artefact that carries the "intentions and norms of cognition and form part of the agency of the activity" (p. 301), and at the same time constrains a person's agency.

Multiple data sources were integrated in the analysis process to develop themes and the lecturer case studies, drawing especially on NVivo. In Phase 1, through inductive reasoning, the research team identified emergent themes (Goetz & LeCompte, 1984) and then refined them as new data were collected. In Phase 2, a deductive analysis process was used to examine our hypothesised three-tier model of software literacy progression. Although we used our model as an initial guide, iterative cycles of analysis required our revisiting the data, the model and the literature to fine-tune the analytical and theorising process and outcomes (Erickson, 2007).

During the 2-year research period, the research team used a collaborative team approach to data analysis to identify patterns, seek explanations for unique findings and ensure collective commitment to emergent findings and their ongoing refinement. Within-case and cross-case analyses were conducted to identify software literacy skills and understandings unique to and common across each discipline.

Findings

The extent to which student software literacy develops, and how

Four key themes emerged from investigating students' perspectives about their learning of software: their general comfort level in engaging with technology; their overall preference for and/or reliance on informal learning strategies in acquiring software skills; their understanding of core affordances and constraints of individual applications; and, a relative absence of critical software literacy among students.

Student comfort level with technologies

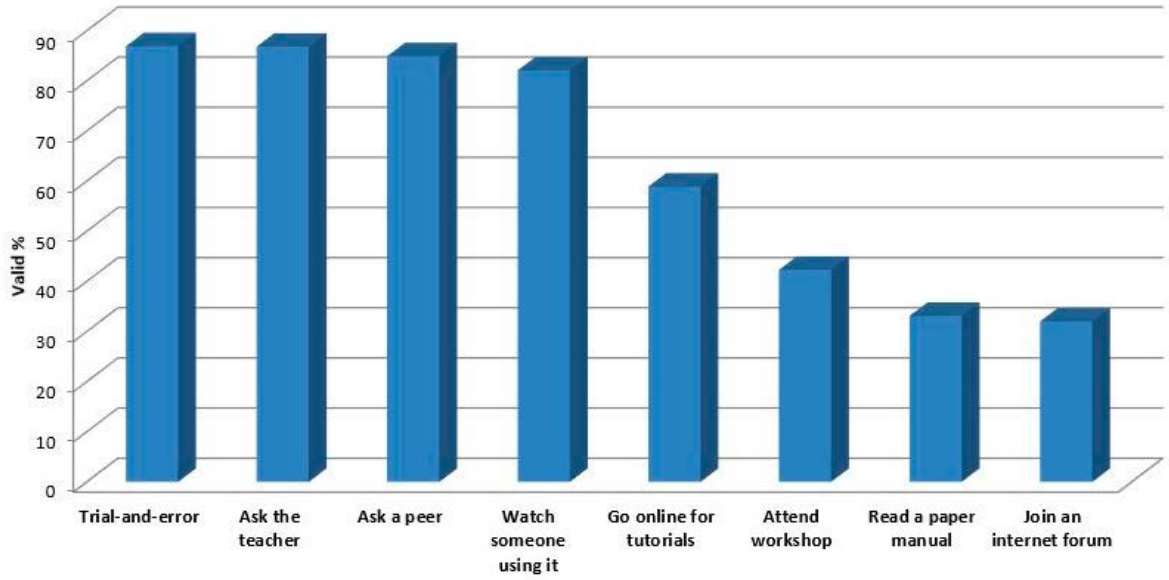
When asked about their general views towards adopting technologies in Phase 1 of the project, 42% of first-year students (*n* = 179) indicated they usually use new technologies when most of their friends do, 30% reported liking new technologies and using them before most people they know do, and another 16% indicated they love new technologies and viewed themselves as among the early adopters to use them. These results illustrate a majority of incoming students (89%) consider themselves early or quite early adopters of new technologies and are comfortable in engaging with new technologies.

Similar results were found in Phase 2 of the project where 38% of students (*n* = 169) reported they usually use technologies when most of their friends do (average across five papers), 35% reported liking new technologies and using them before most people they knew did, and another 18% reported loving new technologies and being among the first to use them. These results illustrate a majority of incoming students (91%) consider themselves early or quite early adopters of new technologies and are comfortable in engaging with new technologies.

Student preference for informal learning strategies in acquiring software skills

Students reported they drew mostly from informal learning resources when acquiring basic skills to use PowerPoint in Phase 1 of the project. Figure 1 shows findings when students were asked to identify "useful", "very useful" and "extremely useful" strategies for learning.

FIGURE 1: **Phase 1: Strategies students used to learn PowerPoint software (collated "useful", "very useful" and "extremely useful")**



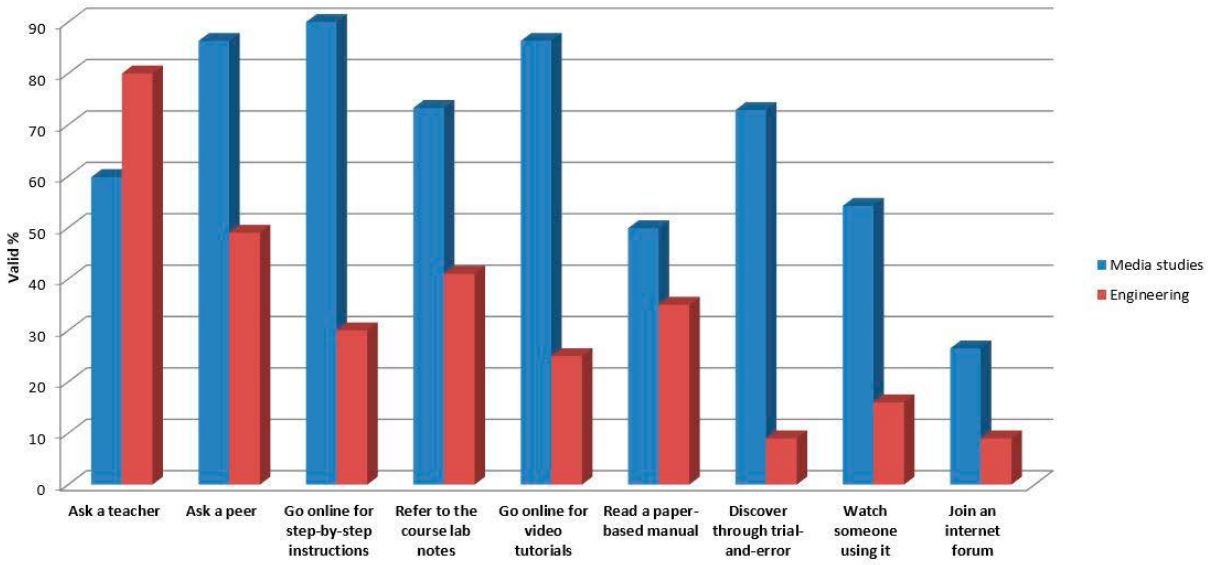
Across both disciplines, the top five strategies for learning PowerPoint were "Discovering through own trial and error" (87%), "Asking an expert" (87%), "Asking a peer" (85%), "Watching someone using it" (82%), and "Going online for tutorials" (59%). These were preferred over other strategies such as "Attending a workshop" (42%), "Reading a paper based manual" (33%) and finally "Joining an internet forum" (32%). Phase 1 focus group students (three groups, 36 students), affirmed these ideas:

> Pretty much [trial and error]. So, if we're given a project or something I'll just figure it out as I go. (First-year media studies student)

> If you don't know how to do something, I'll play around with it for a while, but if it's not like obvious within five or ten minutes, I'm like "sod this, I'll look on Google". (First-year engineering student)

In Phase 2, across both disciplines, similar trends were seen when students were faced with learning discipline-specific software (SolidWorks in engineering, Adobe Creative Suite in media studies) (see Figure 2). The three highly valued strategies (combined "useful", "very useful" and "extremely useful") by media studies students were "Going online to refer to instructions" (91%), "Asking a peer" (86%) and "Going online to refer to YouTube videos" (86%) as useful to their learning of discipline-based software. Engineering students reported "Asking the teacher" (80%), "Asking a peer" (49%) and "Referring to the course or lab notes" (41%) as their preferred strategies. Possible reasons for engineering students' valuing asking their lecturer for help before relying on more informal strategies as compared to media studies students could be due to the perceived complexity of SolidWorks or less experience at school with CAD software in general.

FIGURE 2: **Strategies students used to learn discipline-specific software (collated "useful", "very useful" and "extremely useful")**



In Phase 2, the students in 10 focus groups (61 students in total) outlined a preference for learning at their own pace, sometimes drawing upon "more expert" peers or approaching learning collectively. Their responses most typically centred on using online materials such as YouTube instructional videos:

> Trying to follow a software tutor in class is like watching a YouTube video without pause and rewind. (Second-year media studies student)

Many students highlighted the greater investment in time and attention that the discipline-based software demanded in order to achieve basic competence (and some students explicitly commented that they developed more intensive learning strategies in response). A media studies student considered that discipline-specific software (in his case, Final Cut Pro) was sufficiently complex that work done outside the classroom became an important part of learning:

> … the [lecturers] can give you all the tools but if you're not motivated to do your own experimenting, you're not going to learn the software at all … That's definitely a big part—having the time to actually sit down and play around with it yourself. I mean, you can't expect to just be taught the software—it's something that needs time, you've got to learn it, it doesn't happen overnight. (Second-year media studies student)

This was also the case for engineering students learning to use SolidWorks. A student explained:

> Cause there's so many tiny little individual parts about understanding SolidWorks that you get past a certain point and suddenly you don't know how to mirror a three-dimensional part (for example). (Second-year engineering student)

Regular attendance at formal labs is part of coursework for both disciplines but informal learning strategies were regularly used to supplement formal lecturer-led sessions.

Student understanding of software affordances

In Phase 1, students demonstrated a familiarity with PowerPoint and easily identified its key affordances and constraints. They indicated that as a presentation tool, PowerPoint allowed the embedding of multimedia resources (88%), its in-built templates helped to structure and organise ideas (86%), and information can be incorporated easily into each slide (81%). Our participants also identified the main constraints of PowerPoint: the brevity of information on each slide (68%), the files not containing enough detail for students to understand a lecture (65%), and a tendency for presenters to move too quickly through presentations (63%). In focus group discussions, students expressed confidence in their understanding of and competency in PowerPoint, and quickly applied these to observations and criticisms of their (and other) lecturers' PowerPoint presentation practice.

In comparison, in Phase 2, engineering students were less likely to be familiar with SolidWorks, a proprietary software, prior to tertiary study. In media studies, more students were more likely to be familiar with Adobe Creative Suite or similar applications before university entry.

FIGURE 3: **Media studies students' perception of the affordances in creative software (e.g., Final Cut Pro) that helped them to complete a creative project**

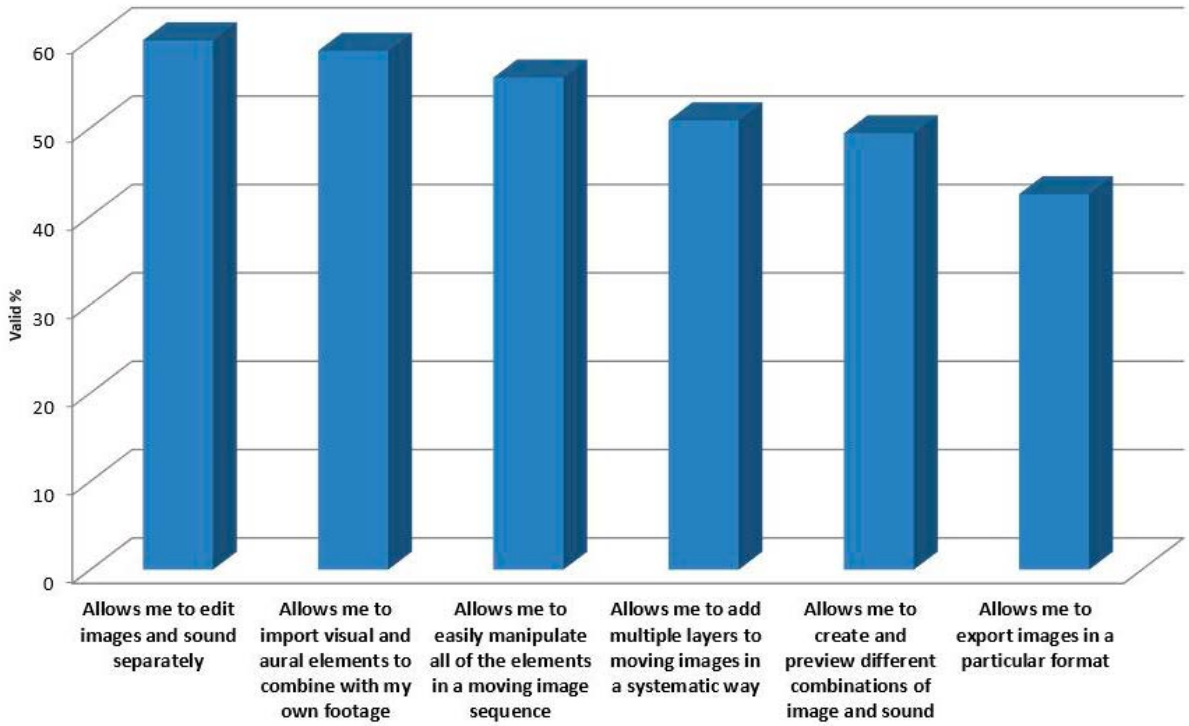
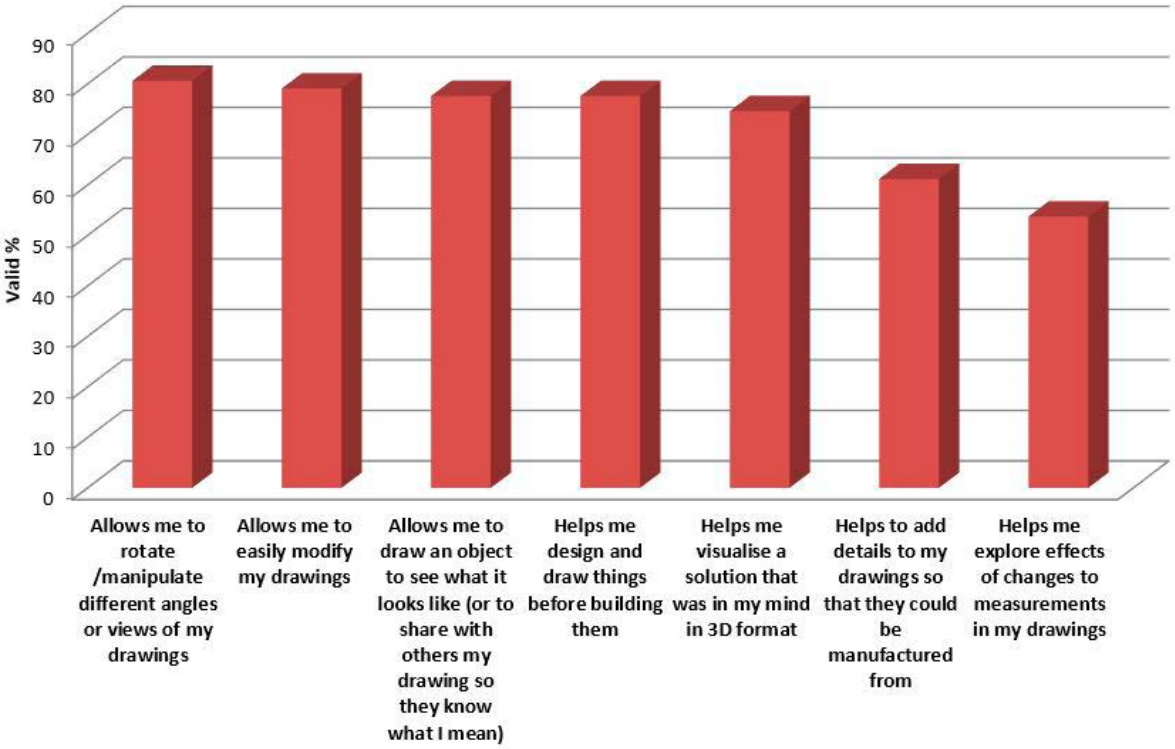


FIGURE 4: **Engineering students' perception of the affordances in SolidWorks that helped them to tackle an engineering design task**



Figures 3 and 4 show that students across the media studies, and engineering papers were able to discern the general affordances of their discipline-based software (tiers 1 to 2 of our hypothesised software literacy model) and to identify the value of these affordances for addressing tasks in their disciplines. For example, the engineering students noted how SolidWorks affords their addressing of engineering design issues—it allows them to rotate and manipulate different views of their drawings (81%), easily modify their drawings (79%), and allows them to visualise their design drawing to share with others (78%) (Figure 4).

Students from both disciplines were aware of the constraints/limitations of software they were learning and commented on areas such as accessibility (e.g., affordability or incompatibility or crashing issues), time and resourcing demands when learning to use the software, and lack of functionality (e.g., wanting to 3D model in video production software, applying/bringing real images into SolidWorks). While students in both disciplines were less likely to identify themselves as "highly proficient" or "expert" in using discipline-specific software at the completion of their course, most nevertheless reported confidence in being able to troubleshoot applications (tier 2 software literacy).

Relative absence of critical literacy among students

A majority of students in Phase 1 used PowerPoint notes in revising for their course (77%). Of the 77%, 56% reported doing extra study to supplement their PowerPoint notes to better understand the lecture content—either through making their own notes (61%), attending the lecture lab or tutorials (61%), or reading the course textbook (60%). Although very few students discussed how PowerPoint shaped their disciplinary knowledge (a key part of critical software literacy), four focus group participants alluded to this by critiquing their peers' reliance on PowerPoint lecture notes, and a common student (mis)assumption that PowerPoint bullet points in and of their own adequately reflected the extent of the disciplinary knowledge presented in the lecture —as in the following representative quotation:

> In PowerPoint, you see a lot of factoids put on the screen rather than actual information. One of the things I noticed the other students were saying that they liked the bullet points. Society as a whole seemed to be heading towards factoid-based learning rather than actual learning. (First-year media studies student)

In relation to discipline-specific software (Phase 2), a majority of students reported shifting in their ability to use the software after learning and using it in the course but had difficulty identifying core disciplinary ideas embedded within software, or felt they were unable to critique the software they were using. Very few students in engineering discussed how SolidWorks shaped their disciplinary knowledge (a key part of software literacy) as was the case also with students in media studies.

How the development of software literacy impacts on the teaching and learning of discipline-specific software in formal tertiary teaching setting

The development of software literacy occurred at various rates across disciplines and was strongly shaped by lecturer teaching approaches, student expectations, and disciplinary assumptions about the need to achieve professional levels of software competency. Many engineering students had little prior experience with SolidWorks, and this resulted in students learning at different rates, with a reliance on asking the lecturer as their preferred strategy for learning the software (see Figure 2). In media studies, more students had prior experience using media editing software (usually at high school) and hence had a higher familiarity with the software, although the diversity in student experiences, knowledge, and background meant lecturers needed to be flexible in their teaching approach.

An engineering lecturer made this observation:

> Some people go through very quickly, gifted students who can follow instructions if [they've used] drawing packages before will pick it up very quickly and they can finish in, I don't know, a tenth of the time.... We accept that there is going to be a big gap between the best and the worst students, just like in maths or anything. You get this big distribution between the best and the worst and I don't think it's any different from any other subject. (Engineering lecturer)

An aspect that appeared to facilitate students' learning of discipline-specific software was students' prior engagement with artefacts or software that had a similar conceptual basis and so provided a pathway for them to engage with new and more advanced software learning—for example, the engineering students who had encountered three-dimensional construction applications with similar sets of affordances to SolidWorks such as ProEngineer, AutoCAD, Star CCM+, Autodesk Inventor and TurboCAD, and media studies students with experience with Photoshop. Across both disciplines, students reported there was an advantage in having prior experience with similar software or a similar interface (45% of media studies survey respondents; 10% of engineering respondents). Transfer of skills and enhanced awareness of functionalities were reported by 15% and 4% of media studies students respectively. Two (3%) engineering students identified transfer of skills as a benefit.

> Actually, most of the editing software are similar to me, if I could use one, then I could use others easily. (Third-year media studies student)

> [Working other similar software] helped to understand and get used to working in 3D on computers. (Second-year engineering student)

Students proposed ways lecturers could approach the teaching of discipline-based software in order to enhance their appreciation of the socioculturally and historically relevant disciplinary ideas embodied within the software. In media studies, students raised the need to understand the broader contextual principles behind the design of a software application for them to better appreciate its relevance and potential applications:

> Like in Final Cut Pro it features words like "bins" and other words and they go back in history to, you know, actual bins that you put film footage into and the cutter will bring them out and cut them. I think that the history of editing and why those terms are used and giving them a bigger picture might just help them realise the terms. [...] it's just that deeper knowledge that's very shallow when you're coming into software if you don't the history of the, you know, industry that goes behind it. (First-year media studies student)

In engineering, for example, teaching the principles of engineering design as well as CAD conventions can enhance student understanding of the potential of the SolidWorks software. In the following comment, a second-year student linked this to working with a real-world case:

> I think what would be cool is if we had case studies or something; just some problems in class we could work through, the teacher could go through, like, "this is something that you may encounter while you're doing CAD, this is how we've gone about it, you could do it your way but this is the procedure we've used". (Second-year engineering student)

To sum up, the student comments highlighted that, in order to cater for diverse student abilities, experience and background, lecturers needed to use a range of strategies (formal and informal) and to be flexible when teaching about and with software to facilitate the students' learning and development of software literacy.

How and in what ways lecturers model attention to, and use different affordances in, discipline-specific software

In Phase 1, lecturers in both disciplines recognised the affordances and constraints (including linearity) of PowerPoint's bullet points and the in-built template structure. When interviewed, lecturers indicated they had a general understanding of PowerPoint's affordances and of best practice in its use. They could articulate a rationale for their own practices and the relevance of these to their discipline content knowledge. They adopted a range of presentation and performance strategies to go beyond the simple presentation of the points detailed on their PowerPoint slides. For example, media studies lecturers extended student learning during PowerPoint-based lectures by incorporating interaction (asking questions or peer/group class activities, using quizzes), sharing of real-life examples to exemplify a point, revisiting earlier points when needed, movement and gesture within the classroom space to capture student attention, and making PowerPoint notes available to students to prompt critical thinking.

However, in Phase 2, there was a difference of opinion between the two disciplines regarding assumptions about the level of software literacy students needed to be a work-ready graduate. In media studies, one lecturer explained that he wanted his students to be able to critique a discipline-specific software (achieve tier 3 software literacy) in relation to other similar software:

> I would love it that they [students] would start [to] become very critical of the pieces of software that they're using and actually start trading ideas about what's the best thing to do, which tool to use in particular instances and … for them to understand and I guess ultimately to throw any piece of software at them and they could start to pick it to pieces really quickly. (Media studies lecturer)

Another media studies lecturer felt that, by providing students with a history of software design, he could encourage them to start asking questions about the affordances of software, and how these affordances fit into the bigger picture software use as part of students developing the ability to critique software:

> [The questions I want the students to be able to ask by the end of the lecture are]—How do people want to use things? How do people want to use the things that you're making? Whether they're media clips or applications or web pages, doesn't matter. What kinds of people do you think are going to be using everything you make? What can you expect them to know? What kind of affordances do you think they're going to be able to detect? (Media studies lecturer)

In engineering, different lecturers articulated the disciplinary assumption that the range of contexts and software applications that engineering graduates would need to engage with is diverse and sufficiently complex such that students would be learning various disciplinary software throughout their careers. However, a basic understanding and development of software literacy up to tier 2 of our software literacy model was viewed to be sufficient in preparing students to be lifelong learners of discipline-based software.

> I do not believe that we're going to teach you [address to students] everything here. It's got nothing to do with that. You will be doing different projects, different software, and different things all through your career and you just have to have a mentality that you'll learn new things. (Engineering lecturer)

> The bottom line is we have [students] who are very good at SolidWorks and when they left [university] they found they weren't doing SolidWorks any more because it's a very expensive software. Companies were just doing sketching by hand and then sub-contract that out to specialists in SolidWorks. So it's not every graduate engineer that has to know this software—[it's enough to] know it exists, and have the basics. Once they [students] go into industry they could be project managers or anything, they know it's there, they know its capability and at least some are really good at it and some are OK, and that's enough for [working in the] industry. (Engineering tutor)

These different disciplinary expectations and assumptions were played out in the teaching approach to discipline-specific software learning. In media studies, for example, observations from the formal lab sessions indicate that lecturers and tutors tended to focus on a general overview of the software's affordances, point out the complexities of the affordances, and attempt to illustrate how these affordances could potentially enable and constrain creativity as students engage and practice using the software. This focus was absent from engineering lectures and labs which tended to point out the general affordances of the software before focusing on the specific tasks and functions that the affordances enabled to address a specific engineering design issue.

Lecturers from both disciplines added that preparing students to engage with the range of possibilities that a discipline-specific software can offer (conceptual and technical) and to apply it to other contexts (tier 3 software literacy) would require strategies beyond formal direct instruction in their coursework, as highlighted in the idea of performative learning in the quotation below.

> … there's tutorials online, there's a whole range of stuff that they [students] can explore for themselves and that's one of the reasons why software teaching is not about being in a classroom saying "this is how you do things"; it's giving them the confidence to open up this world and explore this world on their own and that's how … that's what we do with thinking as well in terms of this whole process. You cannot teach this notion of thinking, actually—they have to get it through practice and performance. So, I see it as a performative idea of learning through action as opposed to learning through telling. (Media studies lecturer)

To sum up, lecturers understood that they could not provide fully immersive training in the applications they introduced at tertiary level. They sought instead to ensure that students had appropriate learning strategies to empower them to understand how to begin to apply those strategies to different contexts, and to explore the software on their own terms.

Student perception of the software literacies that they learnt as part of their tertiary coursework

Student evaluation of their ability to engage with discipline-specific software before and after completing a course indicates some gains in software literacy (see Figures 5 and 6). Based on the categories of "I would need help", "I have the basic skills" (tier 1 of our model), "I can troubleshoot problems" (tier 2) and "I can apply this software" (tier 3), students at the start of their second-year coursework felt they would need help to use course software, or that they would only have the basic skills to use the software.

FIGURE 5: **Student perception of their software literacies *before* attending the course**

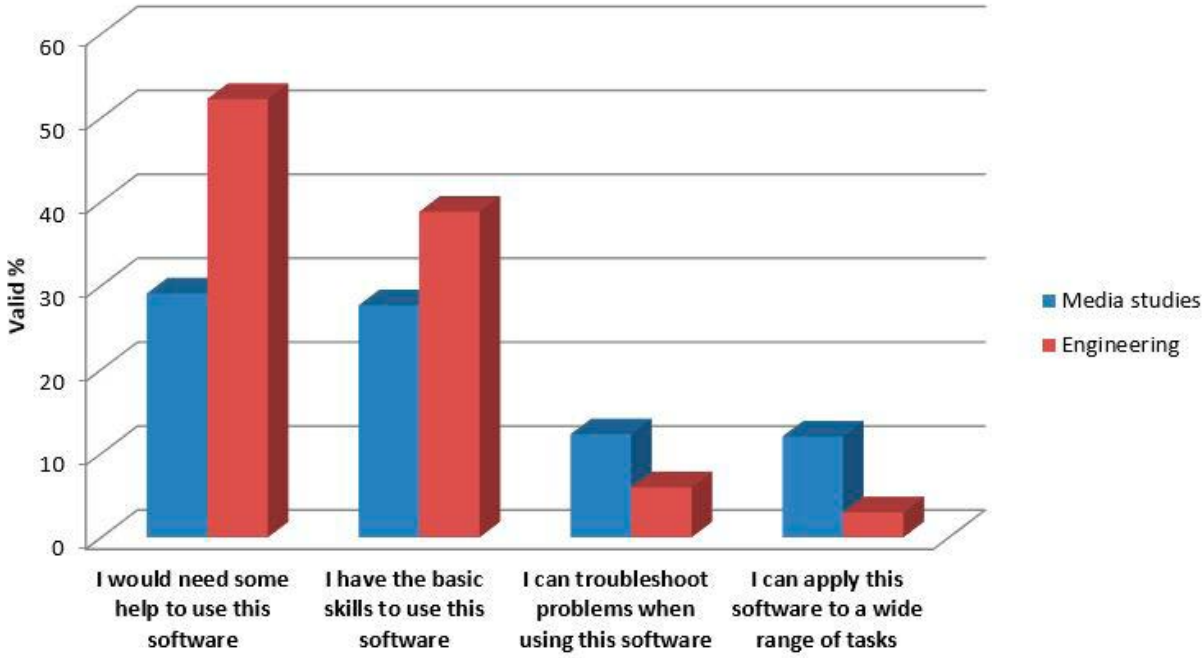
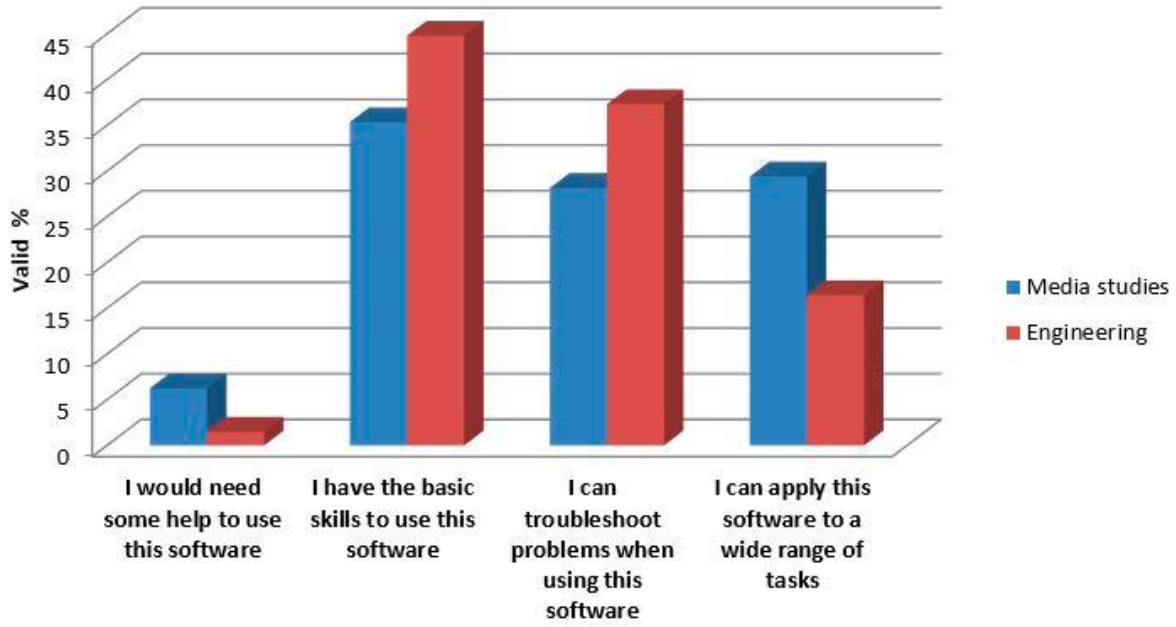


FIGURE 6: **Student perception of their software literacies after attending their course**



These results suggest that formal coursework focused on software learning helped to develop students' software literacy so that nearly all students reported a shift to at least tier 1 (basic ability). After completing a course, about half felt confident enough with the software that they could either troubleshoot problems or apply the software to a wide range in tasks, suggesting a literacy level of tier 2 to 3. This was observed in both engineering and media studies. Very few students report achieving tier 3 of our software literacy model. Triangulation of data sources suggested that the few students who report being at tier 3 were in most cases already competent on entry to the course. This suggests a need for lecturers to consider how they might adapt their teaching to cater to student diversity in software literacy.

How students activate informal learning networks and approaches to support their learning of discipline-specific software

As reported in Figure 2, students regularly drew from informal learning strategies and networks to support their learning of discipline-specific software. These were confirmed by open-ended survey responses and focus group commentary. A representative focus group comment was:

> … most of my learning on SolidWorks has been done by working on it at home or playing around at home, e.g., how to do that, learning from peers and also YouTube videos. Like, if there's no one around and you can't do it, type it into Google, type it into YouTube and hopefully you'll get something and if you don't then get some help. (Third-year engineering student)

Students drew heavily from online sources such as YouTube, Google, and peers, which suggests they have developed an expertise in finding instructional material suited to "their level" of understanding and style of learning. For example, focus group students talked about following particular YouTube channels that provided video demonstrations they understood and or referred to one or two experts' tutorial sites which had (text-based) explanations and instructions they could use to practise and test out their ideas.

How students understand software as an influence on the way they encounter and make sense of disciplinary knowledge

Across both disciplines, as might be expected (see Figure 6), only a few students achieve tier 3 of our software literacy model. The few students who did achieve tier 3 highlighted the general ways their course software helped them consider aspects of disciplinary practice—video production for media studies and construction and design of engineering tasks for engineering. In their open response in the survey and focus group comments, media studies students commented on the freedom and versatility that their discipline-specific software provided (21 out of 54 students). That is, they thought the software enabled them more freedom to create a wider range of aesthetic design possibilities (16 out of 54 students). A representative media studies student quotation reflected the way discipline-specific software such as After Effects enabled him to try out specific disciplinary ideas to make sense of them:

> This software allows me to think more non-linearly, so I can place together footage/audio in new and more interesting ways. (Third-year media studies student)

On the other hand, engineering students reported on the ways SolidWorks enabled them to visualise abstract disciplinary ideas, create and manipulate three-dimensional objects, and communicate their design ideas to others (8 out of 67 students) as indicated in the following third-year engineering student quotation. He appreciated being able to visualise abstract and core disciplinary ideas such as dimensions and conventions in SolidWorks:

> I guess you could say that you can make things in SolidWorks that you can't make in real life. So, […] in SolidWorks you could [drill] a hole that was in a spiral and curve round but then you can't get a drill and drill that. Yeah, just … that was a problem I came into when I was learning because I was just making models as they looked rather than how they could be made. (Third-year engineering student)

Implications for practice

This project aimed at investigating the notion and development of tertiary student software literacy by proposing a three-tier model of development as a response to the ubiquitous but often neglected role that software plays across various sociocultural contexts. The findings support the existence of our hypothesised three-tier software literacy model. However, student development and movement between the tiers was more fluid and flexible than we hypothesised. Student ability to achieve a higher level of software literacy does not necessarily preclude them from needing to revisit earlier levels in contexts where they encounter new but similar software. The tiers are not necessarily distinct but rather the boundaries are permeable. This suggests lecturers should not assume student competency across contexts (e.g.,

informal to formal, from campus to workplace settings). Nonetheless, our model has value as a conceptual tool for practitioners in terms of understanding the role of troubleshooting as an important development stage in learning with and through software.

We see value in students' gaining tier 3 capability, as the ability to critique software is fundamental to understanding that software is not neutral (Manovich, 2008; Fuller, 2008). Instead, software is culturally produced, evolving from particular contexts and needs. Students who are able to transfer a critical understanding of software affordances—who can both use new software and familiar software in new contexts and situations—have the sophisticated understanding needed for considering how software shapes some readily available kinds of knowledge and actions while also constraining other forms of knowledge and actions. Our study found instances of students who had arrived at tier 3 and of students who had not moved beyond tier 2; for example, our finding on student (mis)assumption that the course lecture notes conveyed through PowerPoint's bullet points in and of themselves constitute an adequate unpacking of pertinent disciplinary knowledge. The affordances and constraints that PowerPoint and discipline-based software offers in shaping disciplinary knowledge needs further consideration by students and by lecturers. Lecturers therefore need to be aware of the implications of their choice and modelling of a software application or platform. Findings suggest lecturers would be wise to introduce and help students to develop critical awareness of how such software can inform and shape the students' understanding of disciplinary knowledge and practice.

Even when evidence of students achieving tier 3 exists, it cannot be assumed that they have mastered all facets. Compared with tiers 1 and 2, tier 3 is more complex and therefore difficult to achieve. Its multifaceted demands play out in multiple ways to the extent that we consider the skills and understandings required for tier 3 literacy varies according to the demands of the particular discipline and the forms of knowledge and skills that are valued in that discipline. For example, in the media studies case, critical thinking tends to be seen as a core aspect of creative disciplinary knowledge. Students' development of reflexivity is as important as the development of the capacity to produce a creative product. Tier 3 ability is essential for media studies graduates to be competitive in their profession. For instance, there is a wide range of options in video editing—proprietary applications, less sophisticated open source variations of these, and other options such as online "freemium-based" editing systems. Being able to judge the creative capacities of competing software has implications for the nature of the practice which is being developed, the form and eventual conceptual complexity of media products themselves, and more practical considerations such as budget. In contrast, in the engineering case, the disciplinary emphasis is on the quality of an engineering design. Students are expected to develop digital and software literacy skills as part of 21st century professional learning rather than critique the use of discipline-based software to accomplish a design. For graduates, design efficiency and effectiveness is prioritised over the kinds of creative reflexivity observed in the media studies case study (i.e., the function of a product is more important than the form). We consider tier 3 would be needed by experienced engineers charged with selecting and/or recommending different software as fit for a particular industry's purpose/task. Our findings indicate there is value in each discipline examining how discipline-specific software teaching and learning is positioned in relation to graduate profiles. Software teaching and learning environments where students encounter a range of competing software tools would be needed to raise awareness of the affordances of different discipline-specific software. This in addition to learning a discipline-specific software in-depth can be beneficial to providing students with the breadth and depth of literacies required to engage successfully with software in their discipline.

Multiple learning pathways exist for exploring the affordances of any particular software, both formal and informal. Students prefer informal strategies as a supplement to, and at times above, formal strategies for learning discipline-based software. Lecturers could usefully be informed by, and take advantage of, students' informal repertoire of learning strategies and networks, including their accessing web-based resources and discussing with peers. Lecturers drawing from students' already established informal learning strategies would recognise the relevance of the social and cultural context in shaping effective technology and software engagement.

Our findings challenge current notions and assumptions of the digitally literate generation. Our participants perceived themselves to be competent and confident early adopters of technologies and could identify the general affordances of generic software (PowerPoint) to the extent of critiquing instances of poor PowerPoint practice in their lectures. They were able to recognise discipline-specific software affordances and acknowledge these to be central in their engagement with disciplinary knowledge. However, very few were able to critique how the software might *shape* their disciplinary knowledge. In focus groups centred on discipline-based software, very few students demonstrated critical thinking about the nature and role of the software they were using and most were not able to describe or discuss applications beyond those used in their learning. That is, there was very little evidence of tier 3 software literacy. Students' superficial critique of software challenges current assumptions that today's students, as "digital natives", have developed critical awareness simply through familiarity with and regular use of software. To reiterate, lecturers need to explicitly teach and model software critique if they wish to foster this capacity and/or make this possibility known to students. Vallance and Towndrow (2007) urge educators to adopt an *informed use* approach to using PowerPoint. That is, lecturers encourage students to engage with and think about the content and presentation of PowerPoint slides and how this influences students' interpretations and engagement with disciplinary knowledge.

The diversity of student cohorts and the range of understandings and the variation in familiarity, skills and experience students bring with them to the formal software learning context constitute a further challenge for teaching of and through software. Some students may already have a critical orientation towards software. Our findings indicate an advantage in terms of more advanced software learning for those who have prior experience with other software with a similar conceptual framework. In response to this diversity lecturers could usefully direct time and attention to formatively assessing students' initial software literacy and adapting teaching activities in light of this. This said, our findings indicate there is no single best approach (one size fits all) to teaching discipline-specific software. Lecturers adopting a range of teaching approaches (formal and informal) and being flexible to address diverse learning needs represents a crucial part of supporting student learning. We recognise tensions in terms of time and depth versus breadth of ideas that each lecturer will need to address, and hence reiterate our earlier point for a re-examination of where and how software-based literacies are positioned within each discipline.

This said, we advise caution in the interpretation of our findings because of the need to consider the situated nature of our investigation of discipline-specific software learning. The participants in the study represent a convenient sample of lecturers and students from one educational institutional setting. They were from two distinctly different disciplinary contexts, with distinct and different disciplinary foci and expectations of software teaching and learning. Lecturers were also careful to point out the study only focused on some courses within a programme and all universities have different interpretations of how software teaching and learning can be enacted. Although the findings cannot be generalised, we hope that by providing "rich descriptions" of the context and action (Lincoln & Guba, 1985), readers will be able to draw insights for their own uses from the study.

Conclusion

Software literacy is an essential part of learning and living in the 21st century; something which, we argue, transcends the use of any particular tool and any particular educational, social and cultural context. Software is an increasingly central part of the palette of understandings and skills which comprise the broadening umbrella of digital literacy. As a cultural artefact, software plays a role in reproducing, reinforcing, and augmenting existing cultural practices, as well as generating new practices. Software such as Google, the iOS software in iPhones and iPads, and Microsoft Office software packages are common examples reflecting the extent to which software has become embedded in everyday personal and professional pursuits. The reliance locally and internationally on algorithmically-centred big data for policy and funding decisions is another example of how software tools can influence the way we live. It is therefore desirable and advantageous that graduates have a critical understanding of software to make

more informed choices about their use, can transfer this critical understanding to software they have yet to encounter, and understand that all software has nuanced affordances and limitations. This has implications for education providers, especially universities, in their role in fostering critical thinking and serving as critic and conscience of society. It is crucial to ensure all students and lecturers are supported in teaching and learning processes when these are mediated through and focused on software. As we move to exploit the potential of e-learning platforms, and make use of social media and cloud-based and mobile applications, our research highlights the need for further detailed empirical investigation of software literacy. In tertiary settings, this is needed to ensure equitable and critical learning with and through software.

References

Adams, C. (2006). PowerPoint, habits of mind, and classroom culture. *Journal of Curriculum Studies*, *38*(4), 389–411.

Coolbear, P. (2008). *Research priorities in the tertiary sector. An interview with Dr Peter Coolbear*. Wellington, New Zealand: Teaching and Learning Research Initiative.

Erickson, F. (2007). Ways of seeing video: Toward a phenomenology of viewing minimally edited footage. In R. Goldman, R. Pea, B. Barron, & S. J. Derry, (Eds.), *Video research in the learning sciences* (pp. 145–155). Mahwah, NJ: Lawrence Erlbaum Associates.

Fuller, M. (2008). *Software studies: A lexicon*. Cambridge, MA: The MIT Press.

Gall, M. D., Borg, W. R., & Gall, J. P. (1996). *Educational research: An introduction.* White Plains, NY: Longman.

Goetz, J. P., & LeCompte, M. D. (1984). *Ethnography and qualitative design in educational research*. Orlando, FL: Academic Press.

Jones, C., Ramanau, R., Cross, S., & Healing, G. (2010). Net generation or digital natives: Is there a distinct new generation entering university? *Computers and Education*, *54*(3), 722–732.

Khoo, E., Johnson, E. M., Torrens, R., & Fulton, J. (2011). It only took 2 clicks and he'd lost me: Dimensions of inclusion and exclusion in ICT supported tertiary engineering education. In Y. M. Al-Abdeli & E. Lindsay (Eds.) *22nd Annual Conference for the Australasian Association for Engineering Education* (pp. 166–171). Fremantle, WA, Australia: Engineers Australia.

Kvavik, R. B. (2005). Convenience, communications, and control: How students use technology. In D. Oblinger & J. Oblinger (Eds.), *Educating the Net generation* (pp. 7.1–7.20). Retrieved from http://www.educause.edu/research-and-publications/books/educating-net-generation/convenience-communications-and-control-how-students-use-technology

Lincoln, Y. S., & Guba, E. (1985). *Naturalistic inquiry*. Beverly Hill, CA: Sage.

Livingstone, S., Wijnen, C., Papaioannou, T., Costa, C., & Grandio, M. (2014). Situating media literacy in the changing media environment: critical insights from European research on audiences, In N. Carpentier, K. Christian Schrøder and L. Hallet (Eds.), *Audience transformations: Shifting audience positions in late modernity* (pp. 210–227). Abingdon, England: Routledge.

Manovich, L. (2008) *Software takes command*. Cambridge, MA: The MIT Press.

Maykut, P., & Morehouse, R. (1994). *Beginning qualitative research: A philosophic and practical guide.* London, England: Falmer Press.

Mietenen, R. (2001). Artifact mediation in Dewery and in cultural-historial activity theory. *Mind, Culture, and Activity, 8*, 297–308.

Peeters, J., Backer, F. D., Buffel, T., Kindekens, A., Struyven, K., Zhu, C., & Lombaerts, K. (2014). Adult learners' informal learning experiences in formal education setting. *Journal of Adult Development*, *21*(3), 181–192. doi.10.1007/s10804-014-9190-1

Prensky, M. (2001). Digital natives, digital immigrants. *On the Horizon*, *9*(5), 1–6.

Underwood, J. (2009). *The impact of digital technology*. BECTA. Retrieved from http://publications.becta.org.uk/display.cfm?resID=41343

Vallance, M., & Towndrow, P. A. (2007). Towards the "informed use" of information and communication technology in education: A response to Adams' "PowerPoint, habits of mind, and classroom culture." *Journal of Curriculum Studies*, *39*(2), 219–227. doi:10.1080/00220270601105631

Valtonen, T., Dillon, P., Hacklin, S., & Väisänen, P. (2010). Net generation at social software: Challenging assumptions, clarifying relationships and raising implications for learning. *International Journal of Educational Research*, *49*(6), 210–219. doi:10.1016/j.ijer.2011.03.001

Wertsch, J. V. (1991a). *Voices of the mind: A sociocultural approach to mediated action*. Cambridge, MA: Harvard University Press.

Wertsch, J. V. (1991b). A sociocultural approach to socially shared cognition. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 85–100). Washington, DC: American Psychological Association.

Wertsch, J. V. (1998). *Mind as action.* New York, NY: Oxford University Press.

Summary of Research Community Outputs

Khoo, E. Hight, C., Torrens, R., & Ranger, G. (2015). Tracing software learning and application from formal into informal workplace learning of CAD software. In A. Oo, A. Patel, T. Hilditch & S. Chandran (Eds.), *The Proceedings of the 26th Annual Conference of the Australasian Association for Engineering Education (AAEE2015)* (pp. 515-524). Victoria, Australia: Deakin University.

Khoo, E., Hight, C., Cowie, B., & Torrens, R. (2014*). Copy, cut and paste: How does this shape what we know?* AARE-NZARE 2014 Conference. Brisbane, Australia; 30 November–4 December 2014.

Hight, C., Khoo, E., Cowie, B., & Torrens, R. (2014). Software literacies in the tertiary environment. In B. Hegarty, J. McDonald, & S.-K. Loke (Eds.), *Rhetoric and Reality: Critical perspectives on educational technology* (pp. 410-415). Proceedings ascilite 2014, Dunedin, New Zealand; 23-26th November 2014.

Khoo, E., Hight, C., Torrens, R., & Duke, M. (2014). *"It runs slow and crashes often": Exploring engineering students' software literacy of a computer-aided design software.* AAEE2014 Conference. Wellington, New Zealand; 8-10th December 2014.

Khoo, E., Hight, C., Cowie,B., Torrens. R., & Ferrarelli, L. (2014). Software literacy and student learning in the tertiary environment: PowerPoint and beyond. *Journal of Open, Flexible and Distance Learning, 18*(1), 30–45.

Hight, C., Khoo, E., Cowie, B., & Torrens, R. (July 2014). *Copy, cut and paste: How does this shape what we know?* Presentation at the Tertiary Research in Progress Colloquium IV organised by Ako Aotearoa/ Teaching and Learning Research Initiative Tertiary Research in Progress Colloquium IV, Conference held at Wellington, New Zealand, 10 Jul 2014 - 11 Jul 2014. 2014. See http://akoaotearoa.ac.nz/ako-aotearoa/events/2014-tertiary-research-progress-colloquium-iv

Hight, C., Khoo, E., Cowie, C., & Torrens, R. (2014). *Is software literacy reshaping the 'digital divide*'? Presentation at the "Media Literacy in Digital Age – Cultural, Economic and Political Perspective". Conference held at Centre for Croatian Studies, Zagreb University, 06 Jun 2014 - 07 Jun 2014.

Hight, C., Khoo, E., Cowie, B. & Torrens, R. (December 2013). "The slides are part of the cake": PowerPoint, software literacy and tertiary education. In *Electric Dreams: 30th ascilite Conference 2013 Proceedings* (pp. 379-384). Sydney, NSW, Australia: Macquarie University. Available at http://www.ascilite.org.au/conferences/sydney13/program/proceedings.pdf

Project team

Dr Elaine Khoo is a senior research fellow at the Wilf Malcolm Institute of Educational Research (WMIER), the University of Waikato, with research interests in ICT-based learning environments and online learning pedagogies, with a particular interest in online learning communities, participatory learning cultures and collaborative research contexts. Email: ekhoo@waikato.ac.nz

Dr Craig Hight is an associate professor in Screen and Media Studies at the University of Waikato. His current research focuses on the relationships between digital media technologies and documentary practice, especially the variety of factors shaping online documentary cultures. Email: hight@waikato.ac.nz

Dr Rob Torrens is a lecturer in the School of Engineering at the University of Waikato, with research interests in engineering education, particularly the transition from high school to university and the first-year experience. Email: torrens@waikato.ac.nz

Professor Bronwen Cowie is Director of the Wilf Malcolm Institute of Educational Research (WMIER), the University of Waikato, with research interests in assessment for learning, ICT in science education, and classroom interactions. Email: bcowie@waikato.ac.nz

The project team (from L to R): Craig Hight, Rob Torrens, Elaine Khoo and Bronwen Cowie

Lecturer participants (2013 and 2014)

Associate Professor Adrian Athique, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Ann Hardy, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Lisa Perrott, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Gareth Schott, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Alistair Swale, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Bevin Yeatman, Screen and Media Studies, Faculty of Arts and Social Sciences.

Dr Mike Duke, School of Engineering, Faculty of Science & Engineering.

Dr Mark Lay, Faculty of Science & Engineering.

Dr Chi Kit Au, Faculty of Science & Engineering.

Summer research scholars (2013 and 2014)

Ms Lisabeth Ferrarelli (2013)

Mr Gareth Ranger (2014)