Software literacies in the tertiary environment

Craig Hight
Screen and Media Studies
University of Waikato

Elaine Khoo
Wilf Malcolm Institute of Educational Research
University of Waikato

Bronwen Cowie
Wilf Malcolm Institute of Educational Research
University of Waikato

Rob Torrens
Science and Engineering
University of Waikato

This paper reports on findings from a two-year funded research project exploring software literacy - how it is understood, developed and applied in tertiary teaching-learning contexts and how this understanding serves new learning. The project has looked at MS PowerPoint (a widely used application) and software specific to two disciplines (Media Studies and Engineering). Data was collected through online student surveys and focus groups. Findings revealed that students tend to rely upon informal learning strategies when learning to use all software, and to demonstrate variations in their understanding of software affordances and their ability in applying software to their learning. However, in all cases they were generally not able to critique applications beyond a superficial level, suggesting a need for formal recognition of software literacy as a means to empower students to more critically engage with a variety of / forms of software.

Keywords: software, literacy, teaching and learning, university, New Zealand

Introduction

This paper reports on the initial findings from a two-year (2013-2014) Teaching and Learning Research Initiative funded project exploring how tertiary students develop the understandings and skills needed to use software as forms of *software literacy*. Our framework has been informed by the paradigm of software studies paradigm, our revision of notions of digital natives and digital literacy, and a recognition of the complexities of informal and formal strategies for the learning of new software.

Software studies, a comparatively new field of enquiry that Manovich (2001, 2008, 2013) and others have championed (Johnson, 1997; Fuller, 2003; Fuller, 2008; Kitchin & Dodge, 2011), insists that 'software', operating at the levels of individual applications, platforms and infrastructures, is the dominant cultural technology of our time, an actor integral to many of the social, political and economic practices within contemporary society. A core premise of software studies is the need to move away from seeing software as neutral tools. Instead software users need to develop a critical awareness of how software operates to both 'empower and discipline' us (Kitchin & Dodge, 2011, p. 10-11), contextualising and framing our agency within human-machine assemblages. As we increasingly operate within software culture, our social, economic, political and cultural practices are 'coded'; inseparable from the logics embedded within programming code. Within this paradigm, there is a vital need for detailed empirical research into how software is understood, interpreted, and actually 'performed' by individuals and groups in specific contexts. This project focuses on the literacies associated with software within a tertiary education context. How do both lecturers and students learn, understand and perform different kinds of software, and what are the implications of this for their teaching and learning?

We define software literacy as the expertise involved in selecting, using and critiquing software applications where these are being used to achieve particular goals. Our notion of software literacy is a practice-based schema which anticipates that users can scaffold from acquiring a basic skills in using an application, to appreciating its affordances, and then on to develop an understanding of how software operates to shape and frame knowledge and knowledge generation, and communication and creativity within disciplinary practices.

We view software literacy, then, as encompassing three specific levels of capabilities:

a. a basic functional skill level, enabling the use of a particular application in order to complete a specific set of tasks;
b. an ability to independently problem solve issues faced when using an application for familiar tasks (which includes the ability to draw upon various resources to help solve difficulties); and, ultimately,
c. the ability to critique the application, including being able to apply a similar analysis to a range of software designed for similar purposes - enabling the informed selection of applications and more 'empowered' new software learning.

In these terms, the most 'critically literate' users both develop the ability to identify the affordances of particular software tools and are able to apply and extend their knowledge and use of these and other software tools to a range of new and different purposes and contexts. Ideally, we envisage software literate users being able to recognise, assess and critique the nature and implications of a variety of /forms of software within everyday life. Users may acquire software literacies through a combination of any number of means; through trial and error, learning informally, or training in a more formal or structured way. We assume most people develop proficiency with ubiquitous software packages informally through everyday engagement. Tertiary students are assumed to be able to translate these knowledge and skills into formal settings to complete learning tasks.

We argue that that there is a need to revisit and revise concepts such as information literacy, digital literacy, and related terms (Hegarty et al., 2010; Livingstone et al., 2013). In particular, we need to differentiate between distinct literacies relevant to specific technologies, and to examine the nature of student critique and decision making around which tools might best serve their learning purposes. Labels such as 'digital natives' claim to describe the characteristics of a new generation of learners, capable of operating at 'twitch speed' and able to multitask, imagine, and visualize while communicating in multiple modalities (Prensky, 2001). This term tends to conflate a basic skill with new technologies with broader forms of understanding and the ability to critique aspects of technology-based cultures. We need to unpack this set of assumptions, to more carefully identify the range of skills and other literacies that today's students do (and do not) bring to their tertiary learning. There is emerging evidence that although this generation may be technologically competent, many still lack the basic academic technological literacy skills needed to successfully apply software embedded and enabled technologies effectively to enhance their learning (Kvavik, 2005). A crucial question here is whether, in an environment of universal access to digital tools, the 'digital divide' is being reconfigured as inequalities in software literacies. Recent research indicates that inequalities and marginalisation persist around students' access to, and use of information and knowledge (Bennett, Maton, & Kervin, 2008). Digital inequality is not restricted to just the issue of physical access to software and hardware (Selwyn & Facer, 2007), and given the various forms of investment required in the adoption of ICTs in the tertiary sector, it is imperative to understand how to close the participatory gap for students and ensure that technology is equitably and effectively used (Jenkins, Clinton, Purushotma, Robison, & Weigel, 2006). No studies to date that we know of raise the role of student understanding of how software and its affordances influences knowledge generation and critique, or the influence of formal and informal learning in relation to software.

Research context

In this paper, we report on findings from a two-year study exploring how students develop the understandings and skills needed to use particular pieces of software at the University of Waikato in New Zealand. Two very diverse disciplines of study, engineering and media studies, are being case studied. We first focused on MS PowerPoint (PPT), a widely used application in teaching and learning, and then on the teaching and learning of discipline-specific software (Adobe Creative Suite in Media Studies, and SolidWorks in Engineering). This paper provides an overview of our key findings in relation to student participants for both phases of our research design, which employed online surveys and focus groups. Both disciplines are characterised by high enrolments of students at entry-level (180 and 104 students respectively in 2013) with diverse backgrounds but differ in terms of disciplinary foci and professional pathways. Each discipline uses PPT extensively in lecture sessions, and provides training in specific applications which are positioned very differently in relation to disciplinary knowledge (SolidWorks is compulsory for engineering undergraduates, creative software is an elective within media studies) and professional pathways. The two disciplines use lab-based formal training, and provide resources for additional informal training. An online student survey (179 respondents at first-year level) was used to gauge students' technology and software experience, and student focus groups (36 participants from three papers within each discipline) to gather detailed qualitative data around questions related to software teaching and learning. Analysis of the data was underpinned by sociocultural theory which directed attention to

the interaction between people, the tools they use to achieve particular purposes and the settings in which the interactions occur (Cole & Engestrom, 1993). Emergent themes were identified through a process of inductive reasoning (Braun & Clarke, 2006), then explored further through follow-up group discussions with student cohorts moving through undergraduate papers in each discipline.

Findings

Four key themes emerged from investigating students' perspectives about student learning of software: 1) their general comfort level in engaging with technology; 2) students' overall preference for and/or reliance on informal learning strategies in acquiring software skills; 3) their understanding of core affordances and constraints of individual applications; and, 4) a relative absence of critical software literacy among our participants.

1. Student comfort level with technologies

When asked about their general views towards adopting technologies, 42.1% of first-year students indicated they usually use new technologies when most of their friends do, 30.2% reported liking new technologies and using them before most people they know do, and another 16.4% indicated they love new technologies and are among the early adopters to use them. These results illustrate a majority of incoming students (88.7%) consider themselves early or quite early adopters of new technologies and are comfortable in engaging with new technologies.

2. Student preference for informal learning strategies in acquiring software skills

Students drew mostly from informal learning resources when acquiring basic skills to use PPT. Table 1 shows findings when students were asked to identify 'useful', 'very useful' and 'extremely useful' strategies for learning.

Table 1: Student reported strategies for learning to use PowerPoint

	Have not used (Valid %)	Not useful (Valid %)	Useful (Valid %)	Very useful (Valid %)	Extremely useful (Valid %)	Collated Useful, Very Useful and Ex. Useful (Valid %)
Own trial-and-error	6.6	6.6	22.4	28.3	36.2	86.9
Ask an expert/ teacher	11.3	2	24.5	30.5	31.8	86.8
Ask a friend/ peer	11.8	3.3	32.9	36.2	15.8	84.9
Watch someone using it	15.2	2.6	30.5	29.1	22.5	82.1
Go online for tutorials	39.7	1.3	29.1	18.5	11.3	58.9
Attend workshop	55.8	2	25.9	11.6	4.8	42.3
Read a paper manual	57.6	9.3	23.2	7.3	2.6	33.1
Join an internet forum	59.3	8.7	26	4.7	1.3	32

Those students faced with learning more complicated discipline-specific software (SolidWorks in Engineering, Adobe Creative Suite in Media Studies) recognised these applications demanded a greater investment in time to learn. In focus groups, students outlined a preference for learning at their own pace; sometimes drawing upon 'more expert' peers or approaching learning collectively, but more typically centred on using online materials such as YouTube instruction videos (which notably involved developing an expertise in finding instructional material at 'their level'). Many students highlighted the greater investment in time and attention that these applications demanded in order to achieve basic competence (and some students explicitly commented that they developed more intensive learning strategies in response). In contrast, formal workshops associated with their courses were perceived as less flexible and not sufficiently catering to their learning preferences. For example, course instructors were characterised by one participant as "like a YouTube video without pause and rewind".

3. Student understanding of software affordances

Students demonstrated a familiarity with PPT and easily identified its key affordances and constraints. For example, when asked their views on the opportunities that PPT affords for presenters, students indicated the application allowed the embedding of multimedia resources in a presentation (88.4%), in-built templates helped to structure and organise ideas (85.5%), and affirmed how easily information can be incorporated into slides (81%). Our participants also identified the main *constraints* of PPT: including the brevity of information on each slide (67.6%), PPT files not containing enough detail for students to understand a lecture (65.2%), and a tendency for presenters to move too quickly through presentations (63.2%). In focus group discussions, students expressed confidence in their understanding of and competency in PPT, and quickly applied these to observations and criticisms of their (and other) lecturers' PPT presentation practice. In comparison, those students learning more complicated discipline-specific forms of software (SolidWorks / Adobe Creative Suite) were less likely to be familiar with the applications prior to tertiary study, and less likely to identify themselves as 'highly proficient' or 'expert' in using their applications at the completion of their courses. In focus groups, students relied on more expert peers and group brainstorming to help identify key affordances of the software they used. Students also noted the need for teachers to impart a clear understanding of the potential of each application, to help establish objectives for their own (informal) learning initiatives.

4. Relative absence of critical literacy among students

A majority of students reported using PPT notes in revising for their course (76.8%) while another 56% of these students reported doing extra study to add to their PPT notes to better understand the lecture content (either through making their own notes (60.9%), attending the lecture lab or tutorials (60.5%), or reading the course textbook (59.6%)). Although very few students discussed how PPT shaped their disciplinary knowledge (a key part of critical software literacy), four focus group participants alluded to this by critiquing their peers' reliance on PPT lecture notes, and a common student (mis)assumption that PPT bullet points in and of their own adequately reflected the extent of the knowledge presented in a lecture - as in the following representative quote:

> 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.

In relation to disciplinary-specific software, the majority of students (even those completing their undergraduate degrees) had difficulty identifying core disciplinary ideas embedded within software, or felt themselves competent to be able to critique the software they were using. But they did offer detailed assessments of the manner in which they were taught the software within each discipline, prompted in large part by assumptions about the need to achieve professional levels of competency.

Discussion and conclusion

Our study aimed to understand the extent to which and how undergraduate tertiary students in different disciplines are critically aware of how specific software can impact their learning, with PPT and discipline-specific applications (SolidWorks / Adobe Creative Suite) as cases to understand the emergence of software literacy. Our participants were generally comfortable with engaging with new technologies, with variations only in how quickly they adopted technologies in relation to their peers. They also reported a range of learning strategies that were mostly informal when acquiring basic software skills, particularly the use of trial and error. Both these findings support some assumptions in the 'digital natives' label (Prensky, 2001). Many students identified strategies for trouble-shooting any difficulties they encountered in using software, in particular drawing on peer networks and developing competency in finding, assessing and selecting online resources appropriate to their learning level. In our findings trouble-shooting was clearly 'social' in the sense that students shared their knowledge about strategies and specific resources which were most useful for them (which reinforces and complicates the 'digital natives' term).

Further, most students could successfully identify the key affordances and constraints of PPT use. In relation to discipline-specific applications, students relied more heavily on 'more expert' peers and collective knowledge when attempting to identify affordances of these specialised software. Our participants generally recognised all such applications to be central in their engagement with disciplinary knowledge. However, outside of the occasional focus group response such as in the quote above, student critique of how such applications might *shape* their disciplinary knowledge was surprisingly superficial. In focus groups centred on discipline-specific software, very few students were confident in critiquing the manner in which affordances and interfaces of

applications might have been designed, or could extend their understanding of applications used in their learning

These findings have a number of implications for tertiary teaching and learning. Firstly, teaching and learning of courses involving a focus on software can be informed by and take advantage of students' informal repertoire of learning strategies. Being informed by and drawing from students' already established informal learning strategies recognises the relevant social and cultural contexts that shape effective technology and software engagement and if appropriated accordingly can enhance technology-based pedagogies in the tertiary sector. Secondly, students' superficial critique of PPT revealed that critical awareness does not develop simply from the use of software, rather it needs to be prompted and/or explicitly taught. Even within those disciplines we studied where software is taught in a more formal and focused way, few students demonstrated a critical thinking about the nature and role of the software they were using. Scholars such as Vallance and Towndrow (2007) urge educators to adopt an *informed use* approach to using PPT, that is, lecturers consider how they talk around PPT slides and how they encourage students to engage with and think about the content of the slides influences students' interpretations and engagement with disciplinary knowledge.

There is a more general need here to address two core aspects of the pedagogy around software: the distinctions between how to teach software, and, how to teach *about* software. We argue that the latter builds upon the former: that critical thinking about software first requires 'hands-on' engagement with software in order to develop an understanding of their affordances and potential application for specific tasks. (This draws from a key assumption made within debates over media and information literacies, although we argue for an extension of current definitions of such literacies (Livingstone, 2013) to explicitly includes software as a cultural technology with its own agency). We argue there is also a need to more deliberately scaffold from the teaching of basic software skills to teaching more generally about software as *cultural products* (Manovich, 2013). In conclusion, we view understandings of how software literacy develops and impacts on teaching and learning can lead to insights into the cultural significance of software more generally. Software literacy is an essential part of learning in the twenty-first century, one which we argue transcends the use of any particular tool. This understanding is crucial to ensuring all students and lecturers are better supported in teaching and learning processes that are mediated through and focused on software. As we move into an educational environment looking to exploit the teaching and learning potential of e-learning platforms, social media platform, and cloud-based and mobile applications, our research highlights the need for further detailed empirical investigation in this field.

Acknowledgements

The authors gratefully acknowledge funding support from the Teaching and Learning Research Initiative, New Zealand Council for Educational Research, Wellington, New Zealand.

References

Bennett, S., Maton, K., & Kervin, L. (2008). The 'digital natives' debate: A critical review of the evidence. *British Journal of Educational Technology*, *39*(5), 775–786.

Braun, V. & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101. https://doi.org/10.1191/1478088706qp063oa

Cole, M. & Engeström, Y. (1993). A cultural-historical approach to distributed cognition. In G. Salomon (Ed.), *Distributed cognitions, psychological and educational considerations* (pp.1-46). Cambridge: Cambridge University Press.

Fuller, M. (2003). *Behind the Blip: Essays on the Culture of Software*. New York: Autonomedia.

Fuller, M. (ed.) (2008). S*oftware Studies: A Lexicon*. Cambridge: The MIT Press.

Hegarty, B., Penman, M., Kelly, O., Jeffrey, L., Coburn, D., & McDonald, J. (2010). *Digital Information Literacy: Supported Development of Capability in Tertiary Environments*. Wellington, New Zealand: Ministry of Education. Retrieved from http://www.educationcounts.govt.nz/publications/tertiary_education/80624

Jenkins, H., Clinton, K., Purushotma, R., Robison, A., & Weigel, M. (2006). *Confronting the challenges of participatory culture: Media education for the 21st Century.* Chicago, Il: MacArthur Foundation.

Johnson, S. (1997). *Interface Culture: How New Technology Transforms the Way We Create and Communicate.* New York: HarperCollins Publishers.

Kitchin, R. & Dodge, M. (2011). *Code/Space: Software and Everyday Life*. Cambridge: The MIT Press.

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
Livingstone, S., Wijnen, C. W., Papaioannou, T., Costa, C., & del Mar Grandío, M. (2014). Situating media literacy in the changing media environment: Critical insights from European research on audiences. In N. Carpentier, K. C. Schrøder and L. Hallet (Eds.), *Audience transformations: Shifting audience positions in late modernity*, Routledge Studies in European Communication Research and Education, Vol. 1 (pp. 210–227). NY: Routledge.
Manovich, L. (2001). *The Language of New Media*. Cambridge: The MIT Press.
Manovich, L. (2008). *Software Takes Command* (draft available from http://lab.softwarestudies.com/2008/11/softbook.html).
Manovich, L. (2013). *Software takes command* (International Texts in Critical Media Aesthetics, Vol. 5). NY: Bloomsbury Press. https://doi.org/10.5040/9781472544988
Prensky, M. (2001). Digital natives, digital immigrants. *On the Horizon*, 9(5), 1–6.
Selwyn, N., & Facer, K. (2007). *Beyond the digital divide: Rethinking digital inclusion for the 21st century*. Futurelab. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.3384&rep=rep1&type=pdf
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

Contact author: Craig Hight, hight@waikato.ac.nz

Please cite as: 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. Proceedings ascilite Dunedin 2014* (pp. 410-415). https://doi.org/10.14742/apubs.2014.1244

Note: All published papers are refereed, having undergone a double-blind peer-review process.