Constructivist vs Behaviourist Approaches in Design Computing Education: Implications for the Innovation Economy

Kurt Seemann
Southern Cross University, Coffs Harbour, Australia

This paper presents findings from an action research study over 2 years. The study compared competency standards driven (behaviourist) approaches with discovery (constructivist) approaches in the task of introducing and developing design computing prowess and highly adaptive independent, socially confident, learners. Two first year university student groups were compared. It was found that students exposed to the behaviourist approach initially showed better self confidence in skill tasks but quickly became highly dependent on detailed instruction. They generally displayed relatively little or no initiative to problem solve or meta-learn the software's capability and seemed to have difficulty transferring the use of the software skills to new design tasks without being dependent on new detailed step by step instructions. The constructivist group displayed significant initial anxiety, particularly among most (but not all) mature age students during the first half of their total learning time. However, in the latter stage of the constructivist groups learning period, the vast majority accelerated in their learning to produce advanced use of the software and a high willingness to share their 'discoveries' with peers compared to the first group. They also displayed high confidence to transfer to new design tasks and explore features of the technology independently and socially.

Introduction
Innovation is emerging in Australian technology education settings. This is in part a response to knowledge economy demands world wide. The imperative to develop leaders and flexible people in technologies, science and other areas was initially identified in the foresight study "Matching Science and Technology to Future Needs 2010" (Australian Science and Technology Council & Jack Hilary and Associates 1996) and in the National Innovation Summit recommendations (Innovation Summit Implementation Group 2000). Within this setting it becomes an important task to determine whether the training and development level of new age pre-service technology teachers ought maintain the current post war behaviourist learning tradition that merges trade/industry level training with some 'top up' university level education, or whether the new knowledge and innovation intensive demands of the global economy requires technology pre-service teachers to gain higher order understanding of their profession and be adaptive to a range of new technologies as they develop. There is at the very least a need to establish choice: technical teachers in schools for the trade and manufacturing age and/or skilled professional technology educators who foster understanding among their students in
order for them to thrive in the increasingly technologically and globally driven knowledge age. In such shifting settings it is timely to ask how we ought teach Australia’s new age technology educators and school students?

The new directions in Australian technology education policies and State courses are towards innovation in, albeit modest, recognition of the emerging knowledge economy. This is not restricted to technology as computer education. Innovation and knowledge economy capability requires new ways of thinking to substantially challenge standardised practices in a range of technologies in order to effect innovation: mostly for wealth creation but also for developing greater social and environmental capital. Nonetheless, information technologies are profile enablers of knowledge sharing and formation in most new networks. The idea is to lever knowledge transfer capability and creativity as the export product rather than necessarily the technology 'thing' itself as the product. The new commodities include ideas, innovations, ‘knowledge clusters’ and intellectual property. These intangibles are increasingly dominating new world currencies (Desert Knowledge Australia 2002).

One core capability on the increase in preparing people for the innovation and knowledge age is transfer of understanding and fast positive adaptation to respond to or lead rapid change in any setting (Walker 2000, 2002). Adapting to unfamiliar technologies rapidly, positively and sharing discoveries to facilitate knowledge growth in a team is one key capability to be fostered. It is not surprising that it is the simple things that technology educationists must now scrutinise in their work to determine just how well learning approaches in technology promotes desired and often intangibles qualities. Qualities such as knowledge transfer, team capabilities and communication, rapid and creative adaptation and judicious risk taking are much sought after for life long learning, contribution to productivity in innovation and for simply managing personal challenges through life.

Burns (2000) outlines seven stages for conducting formal action research. They include: problem identification, fact finding, critical review of fact finding to create hypotheses, information gathering, establish procedure, implement action and interpreting findings (Burns 2000, p.447). Action Research, however, is very often conducted by teachers 'on their feet'. The objective being to hone in on improved learning outcomes based on adapting teaching approaches, resources and learning settings and monitoring effects on learning. The feedback from the monitoring process is used to further adapt teaching approaches (actions) until a desired learning outcome is observed. Classroom action research follows similar stages to Burns' above. Typically they include:

- Stage 1: Problem identification
- Stage 2: Plan of action
- Stage 3: Information collection
- Stage 4: Interpretation of information
- Stage 5: Planned adaptation to future action. (MMSD, URL accessed Sept. 2002)

This paper summarises the case findings from an action research study of teaching
design-computing, at an introductory level, to first year technology teacher students at university.

**Problem identification**
A fundamental problem in conventional industry standards driven training is that this essentially behaviourist model of teaching appears to be at opposite ends to the new learning goals of the knowledge and innovation economy which seeks intuitive, risk assessing, creative, innovation driven and socially engaging graduates in technologies. Just as the new economy seeks to develop innovators, people who naturally and critically think outside the box, there remains a highly institutionalised technology training tradition and system, typical in vocational instruction, that rewards and demands standard approaches and outcomes: to perform inside the box. Indeed, the more a learning area is expected to meet standard approaches and outcomes, determined by some established body, the less, it is suggested, will there be inclination for teachers to deliver technical education outside the box of conventions. It is not clear whether outcomes based, standardised technology training yields adaptive learners in design computing compared to constructivist approaches.

McInerney and McInerney sum up the typical differences between learning modeled upon behaviourist compared to constructivist approaches. "Constructivist programs emphasise individual initiative and creative thinking in learning. In many behaviourally based programs there is little if any, scope for individual initiative as students are locked into programmed material to which they have to make a controlled (predictable) response" (McInerney & McInerney 1998, p.121).

Design computing education for the innovation economy needs to extend significantly beyond mere technical software or aesthetic skills. Ability to exploit and use new and different software and hardware (different tools for the same task) and to use new learning approaches so as to enhance communication and team sharing are also to be developed. The ability to have confidence to explore and test what a design computing software can do, between demonstrations of milestone techniques, is important for developing teachers in innovation.

The goal for the action research study was accordingly, to trial two different learning and teaching approaches with essentially equivalent cohorts and determine which kind of teaching approach tends to yield not only more confident design CAD students (pre-service technology teachers), but more adaptive, team/share oriented and efficient learners.

**Plan of action**
A difficulty that may arise in knowledge intensive technologies such a learning new design computing software and hardware, is deciding whether a given 'training standard and perceived technology norm' does more harm than good in fostering adapters and innovators. If we teach technology via one genre (one design computing software and hardware brand to one traditional 'industry' standard way), will the process be inclined to foster adapters, innovators and sharers of knowledge? Will the one genre presentation hinder transfer and innovation?
While performance based approaches and standardised task specific skilling to learning design computing has been the dominant genre in schools, understanding (necessary for transfer and independent adaptive capability) may have been the cost to the perceived benefit. In the standardised content and learning approaches where task performance rather than understanding is the assessed goal, Fosnot (1992) suggests we can expect greater difficulty among students to transfer their skills to new tasks. With learning Mathematics, the example is given where, "Students may be able to perform particular academic activities without really understanding the meaning behind them. It is not unusual to see students skillfully doing mathematical calculations without understanding place-value, and teachers having to re-explain it with the introduction of each new operation because learners never understood it in the first place" (Fosnot 1992).

Table 1

<table>
<thead>
<tr>
<th>BEHAVIOURIST CLASS SETTING (GROUP 1)</th>
<th>CONSTRUCTIVIST CLASS SETTING (GROUP 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment: determined by signing off competency criteria and standards sheets. One project to construct.</td>
<td>Assessment: determined by five set challenges: 5 project briefs which earned higher grades as students showed increasingly ‘inspired’ use of discovered software features and techniques. Each brief exposed the learner to a new feature set in the software at its basic level.</td>
</tr>
<tr>
<td>Learner choice in set tasks: closed. All learners were required to construct the same identical project.</td>
<td>Learner choice in set tasks: open. Tasks one to four required all to complete common project briefs but rewarded for degree of new software features used to add value to the briefs. The 5th project brief was fully learner determined. Marks were awarded for capacity to both integrate and transfer understanding of software gained in projects 1-4, and for extended and inspired use of discovered software features.</td>
</tr>
<tr>
<td>Learning Resources: Each student was required to follow a standardised highly detailed step by step tutorial manual designed specifically to produce the identical project. A basic menu help function was available on screen.</td>
<td>Learning Resources: No manual was issued. A basic menu help function was available on screen.</td>
</tr>
<tr>
<td>Role of Teacher: To demonstrate and facilitate each step presented in the tutorial manual. To help students follow the tutorial manual. To trouble shoot for and with students the typing errors in the manual.</td>
<td>Role of Teacher: To demonstrate and highlight key milestone techniques at least twice per class session for each of the 5 set challenges. To facilitate problem solving and guided discovery of software as required.</td>
</tr>
<tr>
<td>Role of Learner: To follow detailed sequence and instruction (in tutorial manual) in order to show performance against common and identical task specific skills. To individually get competencies signed off to complete the course.</td>
<td>Role of Learner: To actively explore and discover software features to satisfy and add value to each set project brief. To share discoveries and solutions with others.</td>
</tr>
<tr>
<td>Degree of risk taking for learning software: negligible.</td>
<td>Degree of Risk taking for learning software: required and encouraged</td>
</tr>
<tr>
<td>Total allocated face to face learning time: 14 weeks, average 2 hrs p/week.</td>
<td>Total allocated face to face learning time: 6 weeks, average 2 hrs p/week.</td>
</tr>
</tbody>
</table>

The plan of action for determining whether the behaviourist or the constructivist learning models yielded more capable design computing technology teachers for the innovation economy is outlined in the following dot point categories:

- Ability to transfer software features to new tasks
Learning in Technology Education: Challenges for the 21st Century

- Adaptable
- Risk management
- Discovery learning style
- Collaborative style
- Self directed learning
- General level of confidence
- General level of positive disposition
- Overall length of time to achieve.

The key question of this action research study was, "does one approach to teaching design computing yield observable learning benefits to another where innovation, knowledge sharing, risk taking, transfer and independent learning are desired outcomes?"

Common core elements or constants were established within the feasible bounds of a core design computing study unit in a first year university technology teacher education degree. This enabled the main variation between two student cohorts to be the teaching approach used. These common core elements included:

- Same cohort year level (prior learning level)
- Similar gender balance and age ranges (i.e. distribution)
- Same design computing software and hardware
- Same Instructor/lecturer.

Being an Action Research study in a classroom setting, there is an acknowledgement that the findings may be cohort specific rather than universal.

**Information collection**

Overall constructivist approaches appeared to have produced better retention and enthusiasm for the Design Computing software. It was found that students exposed to the behaviourist approach initially showed better self confidence in skill tasks but quickly became highly dependent on detailed instruction. They tended to display little or no initiative to problem solve or meta-learn the software's capability and seemed to have difficulty transferring the use of the software to new situations without being dependent on new detailed step by step instructions. When a typing error existed typically students either did not notice the error until later in the manual sequence or on discovering the error or omission, expressed great anxiety in their ability to proceed. In terms of risk taking, the behaviourist group had typically few students who would spend in-class time exploring and discovering software features for fear of "breaking it" and for fear of "wasting time" away from following the manual. Students who had prior vocational level training or experience in different "design computing software" had great difficulty in learning the given software. Typically they asked for where certain features were and if these were not in the same place or even if done more efficiently in the given software, the fact that it was not what they had been drilled in to learn raised complaints about the software's value. It appeared that students in the behaviourist group who had no former
exposure to similar or same design computing software and hardware, learned the software a little faster and better than those who had such a background. Prior learning, in the context of skills transfer, was generally a negative attribute where the prior learning experiences were likely to have been behaviourist oriented and centred on standardised instruction.

The behaviorist group generally felt highly focused on their own assessment task and rarely took time out to communicate and collaborate with peers. There was a substantially better overall learning outcome in the amount of software skills developed to the time allocated among the constructivist group compared to the behaviourist group.

The constructivist group displayed significant initial anxiety, particularly among most (but not all) mature age students during the first half of their total learning time. One student in this group with a long history and formal VET level training in Design Computing became very distressed and their frustration in not being able to transfer CAD skills to different software and hardware settings and never adequately achieved the learning briefs in some cases. In contrast, there were several younger students who had declared they have never used design computing software before, but accelerated in their learning to produce high value work in a very short period of time, far exceeding the seasoned CAD user in their ability to explore the software and adapt without cries for step by step manuals as aids. This suggests that extensive prior learning from industry training and experience could be an undesirable pre-requisite for innovation-oriented new age technology teacher courses. The complete opposite to the fashion in many States to reinforce such backgrounds as desirable. In the latter stage of the constructivist groups learning period, the vast majority accelerated in their learning to produce advanced use of the software and a high willingness to share their 'discoveries' about the software to peers compared to the first group. They also displayed high confidence to transfer to new situational tasks and explore features of the technology independently and socially.

In summary, it was found that compared to the behaviourist group, the constructivist group generally:

- showed more ability to understand software features by transferring their use to new task situations
- adapted to problems more willingly than blaming the software
- took risks to 'see what happens' rather than stop and seek teacher or 'text book' instructions
- displayed more evidence of new software discoveries in advance of desired minimal software tools to be learned
- demonstrated a higher desire to spend time to share new software features discovered with others rather than more time focused on meeting content task goals
- were substantially more self directed in exploring new use of the software beyond course time, rather than not showing post course interest to further use the software
Learning in Technology Education: Challenges for the 21st Century

- demonstrated a higher level of confidence and collaboration to exploit software features and achieve new effects
- displayed positive disposition towards problems as challenges in learning rather than negative expressions of software/hardware inadequacies
- developed a greater range of design computing and adaptive software capabilities in a shorter overall time.

**Interpretation of information**

Often the most argued position against a constructivist and discovery oriented approach to learning is the despair among both teachers and students in the first stages of learning. Here a high desire occurs for step-by-step instruction for skill establishment. If students and teachers can ride through the initial phase of 'anxiety dips' in a constructivist approach to learning, it is suggested that their meta learning and social learning prowess will far exceed that of the behaviourist group and in better overall learning times. The study suggests that if independent and adaptive graduates in new technologies are desirable for the knowledge and ideas economy, then constructivist and discovery approaches may yield far better results than behaviourist and conventional industry competency training approaches.

**Planned adaptation to future action**

The action research results appeared substantially in favour of the constructivist model of learning for developing innovation oriented technology teachers. In order to verify and improve this finding, the Action Research continued into a third year with the following modifications.

Students were forewarned to 'expect' feeling anxious in the early stages of their learning. This was described as anticipating 'anxiety dips' while they were doing a lot of testing and trialing to get project tasks completed. This appears to have improved the way students manage their learning (greater confidence less blame) compared to the first time the constructivist approach was delivered.

A shared set of reference manuals are planned to be available in the computer lab to facilitate those students struggling to learn independently such as mature age students with a developed background in, for example, conventional CAD. Clearly, a background in an industry course does not appear to be a sufficient basis to issue credit in a constructivist learning program if the objective is to develop independent learners and adaptive graduates.

**Conclusions**

Design computing skills for developing innovators appears to be significantly enhanced through constructivist approaches rather than behaviourist ones, through challenged and facilitated discovery of software characteristics and features rather than through dependence on reporting to standardised content and through promotion of guided discovery learning as well as team and social learning. Such constructivist oriented pedagogy in design computing, it is suggested, not only develops significantly more
independent learners in the software, but also more adaptable and socially confident ones capable of knowledge transfer, exploiting new or unfamiliar technologies and tasks, risk assessment oriented learners and faster all up learning time in software capabilities.

It can be concluded that to produce technology teachers who themselves value developing learners for the knowledge and innovation economy, those teachers are better prepared if nurtured through constructivist experiences in design computing rather than behaviourist oriented learning experiences where typically standardised content and detailed task skills are emphasised. Accordingly, it is suggested that technology teacher education for developing innovators should emphasise the professional higher order end of technology learning which encourages up stream knowledge discovery and knowledge maker rather than the operative end of learning designed to produce standardised approaches reporting to specific tasks and content: the downstream end of the knowledge user.

In the knowledge and innovation economy where being first, adaptive and fast in new ways of thinking, it is argued that preferred teaching and learning in such aspects as design computing is better oriented to the upstream source where teacher and student engage in knowledge development. Where the outcome is to task skill people for the regular maintenance of production, perhaps the downstream approach is more suitable as the standardised knowledge user. The former, however, appears to be significantly better suited for the new knowledge economy and lifestyle demands.

References


Burns, R B 2000, Introduction to research methods (4th edn), Longman, Frenchs Forest, N.S.W.


