An Agile Conversion Masters Degree Programme in Software Development

Karsten Lundqvist, Craig Anslow, Michael Homer, Kris Bubendorfer, Dale Carnegie
Victoria University of Wellington
Wellington, New Zealand
{Karsten.Lundqvist,Craig.Anslow,Michael.Homer,Kris.Bubendorfer,Dale.Carnegie}@vuw.ac.nz

ABSTRACT

The Information and Communications Technology (ICT) industry in New Zealand is growing rapidly. The traditional university courses are preparing an insufficient number of graduates to sustain the growth. Many of the traditional graduate students lack key soft skills that are important in team based software development. This paper reports on the development of a conversion Master of Software Development degree. The students are all graduates with little or no computer science degrees, are taught key programming skills, with a focus on agile development. The programme begins by focusing on individual programming skills through solving problems. Later industrial partners are engaged by providing industrial problems to agile teams of students. The industrial partners are active partners in the agile teams as product owners. By solving the problems, the students develop both technical and non-technical skills while utilizing the skills obtained from previous studies. The results from the first year of the programme are encouraging. A key result is that a high number of students found work in paid IT positions before graduating. The main issue of the first year was introducing too many topics at the same time, over-assessment, not enough communication and contact time, little opportunity for the students to make their own experiences, and learning by making mistakes. The programme has been changed for the next year's cohort to introduce less topics at once, provide time and space for learning, and a redesign of scheduling assessments.

CCS CONCEPTS

•Computing education → Computing education programs;

KEYWORDS

Agile Software Development, Graduate Studies, Individual and Group Work, Internships, Masters Degree, Programming, Soft Skills

ACM Reference format:
Karsten Lundqvist, Craig Anslow, Michael Homer, Kris Bubendorfer, Dale Carnegie. 2018. An Agile Conversion Masters Degree Programme in Software Development. In *Proceedings of The 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, February 21–24, 2018 (SIGCSE '18),* 6 pages.
DOI: 10.1145/3159450.3159540

SIGCSE '18, Baltimore, MD, USA

1 INTRODUCTION

In July 2015 the New Zealand (NZ) government announced the establishment of three information and communications technology (ICT) graduate schools spread across the country. They had determined that the universities of NZ were not preparing sufficient numbers of graduates for the growing ICT industry. Two of the schools were dedicated to Auckland (largest city) and Christchurch (major city in the South Island). The third was later decided to be located in Wellington (middle of the country) through a competitive bidding process. The schools are dedicated to teaching post-graduate qualifications with an emphasis on employer needs, such as communication, critical thinking, business, and enterprise skills. The government has allocated $28.6 million NZD ($\tilde{\$}$20 million USD) over four years to fund the three schools [13].

This paper describes the Master of Software Development (MSwDev) programme provided by our university through the Wellington ICT Graduate School. This paper provides aspects of the background of the programme, the curriculum, and an analysis of student feedback after the first year of providing the programme. The paper concludes with the responding changes to the feedback and recommendations for future iterations.

2 BACKGROUND

Since the beginning of the 1990s there has been a dramatic increase of jobs within the ICT sector in NZ [5] with numerous ICT related positions on the skilled occupations and immediate skill shortage lists used for visa applications into NZ [11, 12]. At our university it is not uncommon to see students leave the Bachelor of Engineering or Bachelor of Science prematurely because they have secured jobs in the local IT industry [6].

Our university was established in 1897 and is therefore a relatively old university in the country. In 2007 an engineering programme was established which built on already existing ICT strengths within the university. Our computer science school offers a three year undergraduate major in computer science and four year undergraduate majors in electronics and computer systems, networking, and software engineering. Various postgraduate degrees are available including masters and PhDs. These postgraduate programmes are primarily research based, with one exception being the Masters in Computer Science degree which is a two year taught degree with a coursework component. The Masters in Computer Science degree has not been a particular popular programme with only a few enrolments each year who are mainly international students.

Our school based the design of the new proposed master programmes on a study of 71 local industries and their expectations of recent IT graduates [17]. It was found that due to the domination

of small and medium sized companies (between 10 and 100 employees), there is a high need for graduate level students, however there are no particular concerns of post-graduate student provision. Remembering that the funding was for post-graduate programmes this was a concern that needed to be addressed. Another major finding of the study was that "soft-skills", such as communication and teamwork, was an especially sought after quality, with several companies reporting that they even employed non-ICT trained students with these soft-skills, expecting to train them in ICT skills.

The problem then was how to satisfy both the post-graduate requirement from the government while providing the local industry with the desired type of students. Two different programmes were developed to target this dichotomy, the Master of Engineering practices (MEP) and the Master of Software Development (MSwDev). The MEP is targeting international graduates who have a desire to work in NZ, providing exposure to group work and physical laboratories, which are often lacking in international study programmes, while also focusing on communication skills. This programme will begin within the next year. In this paper we focus only on the MSwDev programme which we now elaborate upon.

2.1 Master of Software Development – MSwDev

The MSwDev programme began in 2016 with 24 enrolled students. It is a conversion masters degree where the entry requirement is an undergraduate degree that is not in the ICT field. Conversion master degrees in computer science subjects have been especially popular in the UK to provide entry into ICT sector jobs from "unusual" backgrounds [18]. There is a lack of literature that investigates the issues of teaching these compressed programmes. The New Zealand Qualifications Authority had only recently made provisions for such degrees, and this would be the first of its kind in NZ [1]. The target would be students with soft-skills, especially recent graduates from areas where it is difficult to find employment related to their study.

To gain direct entry to the programme, students need some experience with programming fundamentals from their previous education and work. If this was not present a one month intensive programming "boot-camp" course would be available. Successful completion of the boot-camp course would provide entry to the masters programme.

There are three key learning objectives of the programme:

- To bring the students up to a technical level in programming and software engineering so that they can usefully contribute to the ICT industry effectively.
- Give the students skills in team work development, focusing on agile methods, providing opportunities to further develop their soft skills for the ICT sector.
- Provide opportunities for the students to improve their networking capabilities and develop relationships to local businesses and SMEs through industrial led projects and industry involvement through relevant seminars and presentations.

The pedagogical underpinning of the programme is problem based learning (PBL) [3]. In this type of learning there is a minimum number of traditional lectures. Instead the direct teaching is shorter presentations with following engaging problems. Early in the programme there are small designed problems, but through the progression of the courses these problems become close to industrial scale problems, and by the end of the programme the students work on real problems provided by industrial partners, where they function as customers or product owners.

PBL was chosen for several reasons. First of all the method reflects well how the ICT industry operates, especially in agile development. The method also provides opportunities for developing design experiences early in the learning, which is commonly lacking in formal teaching. Through group work exercises the students are "forced" to communicate with each other about the problems, thus give them vocabularies to discuss the design and development, thereby develop effective collaboration skills [15].

PBL was designed to help students to construct flexible knowledge bases, become effective problem-solving skills, become self-directed learners, develop effective collaboration skills, and be motivated to learn [10]. It was therefore the expectation that by using problems, especially from real industry settings, the students would be motivated and experience that they would become closer to attain employable skills. It has been shown that when the problems provide meaningful tasks to the students situation, then PBL will motivate them [9, 14]. It is also important to provide students with problems that are open-ended so that they are controlling the outcome, even at the level of what they are learning [2, 8].

The multiple educational backgrounds of the students was expected to provide both educational challenges and opportunities. For instance the group work situations were expected to provide socio-constructive environments [16]. The students could then use their individual prior knowledge and skills and by bringing these to the group there could emerge a "symbiotic environment" of learning. It was, however, expected that there would be issues with constructive alignment of the courses, especially at the beginning of the course. Each student would be having a unique graduate profile, some with software engineering experiences, others with recent experiences from the "boot camp", and it would therefore be difficult to set assignments that would reflect learning outcomes exactly at the level that each student would need in the context of the learning outcomes, a key issue of constructive alignment [4]. To overcome this issue the beginning courses were designed with "layered" exercises where the beginners would have achievable exercises, and then provide extra challenges for students with special interests or skills in the area. The difference of software engineering skills and knowledge of students would be expected to be aligned through the progression of the programme, therefore the student variety would be expected to grow towards opportunities rather than challenges within the educational setting.

3 PROGRAMME STRUCTURE

Throughout the programme the students work full-time (7 hours per day, 4 days per week with Fridays available for coursework and further learning) in a work environment that aims to bring an ICT industrial studio experience. The purpose built space is in a downtown building (located at our city campus) with inspiring views of the city and water front bay area. The space is dominated by an open environment with flexible setup opportunities. All tables and chairs have wheels and can be moved around. Walls have integrated whiteboards and notice boards for sharing ideas. There

Course	Name	Credits	Weeks	Assessment Weightings
SWEN131	Programming for Software Development	15	4	60% Group project, 20% Individual project, 20% Reflective essay
SWEN501	Professional Programming Skills	60	4	90% eight individual programming assignments, 10% three group presentations
SWEN502	Software Development Studio I	45	9	80% four individual programming assignments, 20% group project
SWEN503	Software Development Studio II	45	9	20% group project, 80% two group projects (each 20% group & 80% individual)
SWEN505	Professional Seminar	15	18	75% essay, CV building, hackathon, 25% participation
SWEN589	Industry Research & Development Project	60	14	20% journal log book, 20% oral presentation, 60% final report

Table 1: MSwDev Courses, including number of credits, duration in terms of number of weeks, and assessment weightings. The courses happen in sequential order. SWEN505 (seminar course) runs concurrently with SWEN502 and SWEN503.

are three "break out" rooms for smaller groups, and several movable whiteboards and televisions are provided for presentations. There is a kitchen with access to tea & coffee, and fridges for storing food. Each student also has a personal locker for storage.

The programme was designed with six different courses (see Table 1). Due to the compressed nature of the programme the courses do not follow our standard trimester timetable used at our university. The courses are delivered in sequential order except for SWEN505 which is running in parallel to two other courses: SWEN502 and SWEN503. SWEN131 is formally not part of the masters degree but an entry requirement for students with no prior experience of programming. SWEN131 runs in the (southern hemisphere) winter term break in June/July, which allows students to take the course straight after their courses but before the actual formal programme begins.

SWEN131 introduced the students to fundamental programming constructs. It began with two days of Lego Mindstorms, where the students learned about decision marking and actions. After these two days the rest of the course introduced programming using Java. This included input/output, syntax and variables, conditionals, loops (while, for, foreach), and ArrayLists. The more advanced topics of this course were classes and objects with an introduction to object-oriented design, types, interfaces, GUIs (with a domain specific UI library), and event programming. In the beginning the coursework was individual, but in the two last weeks of the course the students started working in groups on projects. In the first week of the course a marked individual project began on the first day of Java, in the hope of providing students a sense of reward for picking up the elementary aspects. Immediate feedback from students, however, indicated that this developed a stressful environment for them, and took away opportunities for learning by making mistakes, while the students in this programme did not require the award of marks to motivate them to work. The following courses of the programme thus took a slower pace at introducing marked work.

In SWEN501 the first three weeks were dedicated to learning more advanced programming concepts and techniques. Object-oriented design was expanded with concepts such as inheritance, encapsulation, overloading, and overriding. Version control using Git, along with unit testing, were incorporated to prepare the students for upcoming group work. Collections, generics and exceptions were also taught. The first marked coursework was a development of a textual "bug world" simulating a small ecosystem of bugs searching for food. Sorting algorithms were then introduced and the students studied complexity issues and created tests to underpin a short report. JavaFX was then used to teach professional GUI design and the students developed a graphical 'bug world." The last week was a dedicated group project where the students developed a graphical application simulating Conway's game of life [7]. At the end of this course the students were at an approximate level of a first year undergraduate computer science degree compared with students on our standard programme.

The aim of SWEN502 was to further develop software development skills and build up an understanding of related concepts and techniques. Eight different topics were offered in three week blocks of teaching, with different topics taught in morning and afternoon sessions. Each student needed to take six topics in total. The topics were Web Application Development, Cloud Application Development, Software Testing, Mobile Application Development, Human Computer Interaction, Cybersecurity, Algorithms and Computer Science, and Artificial Intelligence. Each of the topics included its own exercises and assessments.

SWEN503 had two primary aims: to give the students experience in team work, especially in agile development, and introduce them to real-life software development. The course had three iterations of projects provided by industry partners. Each group had an academic supervisor and an industrial partner. The learning was self-directed, guided by the needs of the groups to achieve the goals of the projects. Each project was assessed using a group presentation and individual reports. The assessment was put in place due to university regulations that overall assessment cannot be more than 20% group-based.

SWEN505 was based on a series of seminars (9 in total) provided by local and international industry and government experts. Many of these seminars provided opportunities to improve employability through CV building and interview training. Other seminars were inspirational with introductions to current trends and technologies. The course was assessed through CV building, an essay, and a hackathon day where the students developed a business idea.

SWEN589 comprises of a substantial individual project. There were three different strands: industry internship, entrepreneurial development, and academic research. The internships had to be paid positions to work on a project at a company. The entrepreneurial development started with a taught component of entrepreneurial theory and with support the students would develop a business idea with a prototype. The academic research strand was available for students who wanted to pursue a PhD in the future. None of the students had the academic research desire though, but the other two strands were equally popular.

4 DISCUSSION

19 students enrolled in SWEN131, with a total of 25 students at the start of SWEN501 including some who had another equivalent background. During SWEN501 two students stopped, and a further one during SWEN502, but the 22 remaining students continued to the end of SWEN589. The five different industrial projects in SWEN503 and the resulting solutions were well received by the companies. One resulted in an offered paid internship based on the developed prototype, and another company integrated the developed solution into their own online e-Learning solution. All of the project groups managed to produce results that were in line with the problems provided. The students have proven to be employable in the local industry. Before graduation over 75% had accepted job offers or had started in paid positions. One student received funding for their entrepreneurial project through a business incubator run at the National Museum of NZ (Te Papa). The results of the students, both academically and through employability, indicates that the programme is fundamentally achieving the initial goals.

4.1 Teaching Experience

The teaching style of the programme is different from our undergraduate and traditional taught postgraduate programmes. It is useful to reflect on the differences from a teaching perspective.

The traditional use of hour-long lectures is ingrained in teaching practices, but it has been a liberating experience to develop the material in shorter presentations with an emphasis on problems and how to solve them. The students seemed highly motivated by this approach. The only real issue with the approach was that too many of the problems were assessed, and therefore the students were constantly being assessed. This created problems especially in the beginning of the programme because the students did not have time to absorb the new material and construct their own knowledge and skills without being scared of making mistakes.

Interacting with the students has been a joy. Spending full days of teaching, supervising, helping and supporting results in a closer teaching experience learning needs can be discovered as they emerge. Often content was changed because special needs and common problems were identified through the learning processes. The students interacted more freely with the teaching team than in a normal lecturing situation. They asked questions more freely and there was a lot of peer-learning happening as well, especially in the group work situations, but also in the individual assignments.

One partial obstacle in teaching students from such diverse academic backgrounds is that many of them begin with what appears, from a computer-science perspective, to be a fairly shallow concept of "understanding." Some students will see themselves as understanding a topic when they can describe it, rephrase it, and contextualize it, rather than when they can apply it. These are useful skills for communication and part of what we want to develop in all students, but especially early on represented an obstacle to learning when a student was stuck on a problem as they sought individual assistance from the lecturer. Following a discussion with the lecturer, a student who was skilled in this fashion could talk about and explain a problem or solution "in their own words", even including other examples, which gave an appearance that the problem they had been experiencing had been solved. Both student and instructor might leave the interaction believing so, but the student would then find themselves stuck in exactly the same place they had begun. While they could describe the problem, cause, and solution, they did not understand it at the level of applying that knowledge in practice. We learned to insist on correcting the problem at hand practically while present, as well as theoretically, even for students who preferred to write the code by themselves after discussion. In this way, at least some progress was made, avoiding frustration, and deeper or further misunderstandings could be detected.

Being a new programme the teaching has needed a good measure of flexibility. It has not been without problems, especially within the SWEN502 and SWEN503 courses. The arrangement of topics where students did two topics at the same time in SWEN502 was problematic. The students had difficulties prioritizing the various problems they were given, resulting in a stressful time. Moreover the normal flow of delivery with short presentations followed by problems and assignments in SWEN131 and SWEN501 was difficult to achieve because of the limited time for each session.

In SWEN503 there were legal issues with intellectual property rights. The deeds used at our university for dealing with student's intellectual property rights when working on industrially provided projects has been designed for longer running projects, and late in the process this approach was deemed inappropriate for the three week long projects that would be used in SWEN503. Several projects were lost because of IP rights that some companies wanted to retain. The solution was to only use open source projects or projects where the industrial partner did not need such an agreement. In SWEN503 it was also a problem that the assessment had to be based 80% on individual assessments. With only anecdotal evidence present it was felt that there was a skew in the marks towards good report writers. Good coders and team workers, who did not have the same training in report writing as someone with an undergraduate degree in for example English or creative writing, seemingly were challenged more by this assessment regime.

The planning of seminars run in SWEN505 was done by administrative staff within the ICT graduate school. There were issues of planning the timing of these seminars. Teaching is supposed to happen all day, yet seminars were planned at different days and times of the week. This was disruptive for the teaching staff's planning.

4.2 Student Feedback

We obtained qualitative feedback from the students through a series of focus groups at the end of the programme. We ran four focus groups each lasting an hour with either five or six participants. Students spent time writing their thoughts down (positive and negative) into six broad themes on post-it notes and then they were discussed in the group. We now discuss the main findings from the themes that evolved from the analysis of the focus groups.

Programme Structure. Many of the students liked the practical nature of the programme as it was hands on and helped them to develop their programming, presentation, soft skills, and team work skills. Many felt that SWEN131 was intense, great as there were many challenges, and good preparation. Students liked that they could select different topics in SWEN502 and there was a variety of topics as they could tailor their subjects to their preferences. The topics they did like included Agile, Version Control, Security,

and Testing. Students liked the industry talks in SWEN505 as this helped reinforce the ideas they were learning in class and have a greater appreciation of the tools and techniques being used by practitioners. Some also appreciated the workshops to improve their CVs and interview techniques. Some liked that they could choose either an industry placement or entrepreneurship project for SWEN589. Many commented that the speed of delivery and order of courses was about right for the programme.

There were some ideas suggested to improve the program structure. The first two courses (SWEN131/501) were too intense compared with the remainder of the programme and not all students were at the same level at the end of these courses. Students would have liked more exposure to other programming languages and topics in particular Python, and more details on Java, JavaScript, and HTML/CSS. Some struggled with learning Git and would have liked more time. Essentially all students were craving more in depth knowledge on many of the topics in SWEN501 (e.g theory of OO programming) and SWEN502 (e.g. software architecture, design patterns). There was no topic on databases which affected the students as later in the course they struggled to communicate effectively about the vocabulary for databases. Some would have preferred not to have a choice of topics for SWEN502 instead everyone cover all topics. Some commented that it would have been useful to explain clearly how the different topics fitted together. Due to the nature of being the first instance of the programme some students struggled with last minute changes to the structure. Some felt SWEN505 was a bit disorganized and that there were too many presentations. We exposed students to industry projects, however some students worked on internal projects but some wanted more exposure to industrial settings. Some wanted to learn more industry relevant skills and wanted more time to learn the technical practices to become more confident on the industry projects. The internships and entrepreneurship strands needed to be planned more effectively so students knew what was happening earlier. The entrepreneur instructor was based in another city and had to commute which made it difficult to engage with them, hence it would have been preferred to have someone locally based. One student felt that the programme was too theoretical and that it needed to be more practical, however this may have been a case of misunderstanding the expectations of the programme.

Group Work vs. Individual Work. Students really liked working in groups (including team size) especially when it came to different projects in SWEN503. They found Agile methods in particular Scrum and tools (e.g. Trello) to be stimulating and exciting, as it helped them to learn from their peers. Some would have liked Agile to be taught earlier. Due to working in software development teams for the first time students appreciated that their team work skills improved over time and that a great rapport among the students developed. Some felt there was a good balance between individual and group work and the programme encouraged self learning.

A couple of students would have preferred smaller teams for the group work (< 5/6). Some groups had dominating characters which made it hard to ensure that the focus was on learning for everyone. For individual work some would have liked to see more challenging exercises for the stronger students and more effort for those needing more practice on certain programming tasks. Some felt we should have encouraged more independent learning as opposed to lots of group work as they did not like to learn from peers. Given the nature of IT projects in the real world we felt it necessary to have opportunities for group work as that was one of the findings as part of the requirements for the programme. For the entrepreneurship project some found it very hard to work on their startup idea on their own and would have liked more support.

Projects. All students liked the industry projects (including variety) for SWEN503 and SWEN589 as they were interesting, gave them some perspective on real world IT projects, and that they got experience working in a professional self managed team. They all found the industry people who they worked with very helpful. The entrepreneurship project was flexible and helpful as it allowed students to develop their ideas with professional coaching. The students appreciated there was a dedicated team in the ICT graduate school to help them to connect with industry people when required which also helped provide good networking opportunities.

There were some problems with some of the industry projects as they were not planned adequately and that the project sponsors were ill prepared for some of the questions raised by the students. There was an expectation that all students would have an industry internship, but some companies pulled out at the last minute so students worked on internal university projects instead. Ideally we would like to have had industry internships for all. Some students would have liked to be more strongly connected to the startup community and their industry projects to get a better perspective, but this aspect is highly dependent on the company culture. Some students wanted to do an industry internship and an entrepreneurship project, but this was not possible due to time constraints.

Teaching and Supervision. Most of the students found all the lecturers and tutor to be approachable, were good teachers and supervisors, and that there was support if they needed it. A couple of the students struggled as they found it hard to interpret what information was being conveyed and did not ask enough questions. We found that sometimes it was hard for us to communicate to some students due to them being reserved and not asking enough questions. Some felt that the lecturers went the extra miles to impart knowledge while others zoomed passed and with high expectations. During SWEN503 and SWEN589 many students would have liked more supervision from the lecturers. The feedback and supervision they did get on the programme was excellent. They just needed more contact time (e.g. 9–5pm), one on one sessions (e.g. tutor), pastoral care, and in person technical support. Feedback was mainly positive, but some students wanted to know what they could improve upon (e.g. code quality, software design). The entrepreneurship students found the lecturer to be very approachable and considerate, but some would have liked more supervision.

Assessment. Due to the nature of this kind of programme we had to figure out a different kind of assessment for the courses compared with our standard programme at undergraduate level. All of the students struggled with understanding what was involved in terms of assessment and criteria for many of the assignments. This also included communicating the assessment criteria effectively. Some felt the assessment criteria varied widely among the lecturers as some emphasized more on coding while others on the process. Many of the students wanted clear and consistent information about what questions a report and assignment should answer. A couple of students commented that the assessments built

understanding and focused on learning materials and concepts as opposed to just focusing on competency. Some would have liked to seen more assessment of individual programming skills and less on the reflective reports. Some would have liked more quality and academic rigour when assessing the reports, and feedback reported more efficiently. The schedule for SWEN505 should have been finalized earlier including what was involved with the assessment. Due to organizing the logistics for the voluntary speakers for SWEN505 this was somewhat troublesome to organize. For all the courses (in particular SWEN505 and SWEN589) students would have liked the assessment criteria explained earlier as opposed to last minute, but due to the nature of the programme we tried to be as flexible with what we had intended to teach hence the assessment criteria became adaptable based on what we taught and meeting the abilities of the students compared with our expectations. The assessment for SWEN589 in particular the entrepreneurship was not easy for students to interpret and needed more clarification.

Facilities, Equipment, Scholarships. Each student was allocated a laptop for use during the programme, and they all liked that they could keep their data on an individual machine and use wherever. All students appreciated they had a new facility (including equipment: whiteboards, large dislpays, breakout rooms, bean bags, coffee/tea) and was located downtown. Students liked that there were scholarships for the entrepreneurship projects.

4.3 Changes to the Programme

Based on the teaching experience and feedback numerous changes have been planned for the second cohort on the programme.

SWEN131 and SWEN501 will run with only minor changes. One of the group assessed assignments for SWEN131 will become an informal assignment for learning while the others will be reduced in size from the previous assignments. Otherwise these courses are repeated using the same learning objectives and only minor changes made to the assessments. Additional material will be added to SWEN501 to help with communicating software designs more effectively with documentation and diagrams.

SWEN502 and SWEN503 have had significantly more changes. For SWEN502 the teaching schedule will follow an all-day teaching pattern as used in the previous modules. The topics will be taught in weekly blocks with only one topic a day. The topics have also changed to Databases, Cybersecurity, Human Computer Interaction, Web Apps and Mobile Apps. There will be a one-week project in the middle of the course and a two-week industrially-focused group project at the end of the course.

SWEN503 will start with four weeks of taught topics, similar to SWEN502. The topics will be Other Programming Languages, Cloud Apps, Agile Methods, and Artificial Intelligence with Data Mining. These have been introduced because the Christmas holidays are in the middle of the course, and it was disruptive to the flow of the course. The students will now get more "tools" before starting the projects and it is the hope that there can be a wider range of industrial projects. After these topics will follow two iterations of industrial projects in agile groups.

SWEN505 seminars will be exclusively scheduled on Fridays, where students are more flexible with their time. More topics will be covered and excellent previous presenters will be invited back.

5 CONCLUSIONS

With the growth of the ICT industry in NZ there has been a need to increase the number of graduates to meet this demand. In this paper we have reported on the development of a conversion Master of Software Development degree. The programme focuses on key programming skills, agile development, individual programming skills through to solving problems. Companies are engaged in the programme who provide industrial problems to agile teams of students and are active as product owners. The results from the first year of the programme have been encouraging. A number of students found work in paid IT positions before graduating. The main issue of the first year was introducing too many topics at the same time, over-assessment, not enough communication and contact time, little opportunity for the students to make their own experiences, and learning by making mistakes. The programme has been changed for the next cohort to introduce less topics at the same time, provide the students with time for learning, and a redesign of scheduling the assessments. We expect the programme to make improvements each year at to grow to a much larger intake.

REFERENCES

[1] New Zealand Qualifications Authority. 2016. The New Zealand Qualifications Framework. http://www.nzqa.govt.nz/assets/Studying-in-NZ/New-Zealand-Qualification-Framework/requirements-nzqf.pdf Accessed 02-08-17. (2016).

[2] Albert Bandura. 1997. *Self-efficacy: The exercise of control.* WH Freeman/Times Books/Henry Holt & Co, New York.

[3] Howard S Barrows. 1986. A taxonomy of problem-based learning methods. *Medical education* 20, 6 (1986), 481–486.

[4] John Biggs. 1996. Enhancing teaching through constructive alignment. *Higher Education* 32, 3 (01 Oct 1996), 347–364.

[5] Stephen B Blumenfeld and Glen Thickett. 2003. Surfing the knowledge wave: The impact of information and communication technology (ICT) on'decent work' in New Zealand. *New Zealand Journal of Employment Relations* 28, 1 (2003), 1.

[6] Dale A Carnegie, Peter Andreae, Craig A Watterson, and Kris Bubendorfer. 2016. The development of postgraduate ICT programmes: For an industry that does not want traditional postgraduate students. In *Global Engineering Education Conference (EDUCON)*. IEEE, Abu Dhabi, UAE, 702–708.

[7] John Conway. 1970. The game of life. *Scientific American* 223, 4 (1970), 4.

[8] Carol S Dweck. 2000. *Self-theories: Their role in motivation, personality, and development.* Psychology Press, Philadelphia.

[9] Michel Ferrari and Ram Mahalingam. 1998. Peersonal cognitive development and its implications for teaching and learning. *Educational Psychologist* 33, 1 (1998), 35–44.

[10] Cindy E Hmelo-Silver. 2004. Problem-based learning: What and how do students learn? *Educational psychology review* 16, 3 (2004), 235–266.

[11] New Zealand Immigration. 2017. Appendix 6 - List of Skilled Occupations. https://www.immigration.govt.nz/opsmanual/35165.htm?_ga=2.202136152.839163081.1501633296-1032907016.1501633296 Accessed 02-08-17. (2017).

[12] New Zealand Immigration. 2017. Immediate Skill Shortage List. http://skillshortages.immigration.govt.nz/immediate-skill-shortage-list.pdf Accessed 02-08-17. (2017).

[13] Steven Joyce. 2015. ICT Graduate Schools announced in Auckland and Christchurch. https://www.beehive.govt.nz/release/ict-graduate-schools-announced-auckland-and-christchurch Accessed 02-08-17. (2015).

[14] A. N. Leontiev. 1978. *Activity, Consciousness, and Personality.* Prentice-Hall, Englewood Cliffs, Nj.

[15] Julie E Mills, David F Treagust, and others. 2003. Engineering education fi!? Is problem-based or project-based learning the answer. *Australasian journal of engineering education* 3, 2 (2003), 2–16.

[16] Leslie P Steffe and Jerry Edward Gale. 1995. *Constructivism in education.* Lawrence Erlbaum, Hillsdale, NJ.

[17] Matt Stevens and Richard Norman. 2016. Industry Expectations of Soft Skills in IT Graduates: A Regional Survey. In *Australasian Computer Science Week Multiconference (ACSW)*. ACM, 13:1–13:9.

[18] JB Thomspson and Helen M Edwards. 1999. Providing new graduate opportunities in software engineering experiences with a UK Master's Level Conversion Course. In *Software Engineering Education and Training*. IEEE, 50–61.