A Model of Sustainable Ecosystem for Software Development, Software Business and Music Education

by

CHENG, Lee

A Thesis Submitted to

The Hong Kong Institute of Education

in Partial Fulfillment of the Requirement for

the Degree of Doctor of Philosophy

October 2013

STATEMENT OF ORIGINALITY

I, Lee Cheng, hereby declare that I am the sole author of the thesis and the material presented in this thesis is my original work except those indicated in the acknowledgement. I further declare that I have followed the Institute's policies and regulations on Academic Honesty, Copy Right and Plagiarism in writing the Thesis and no material in this thesis has been published or submitted for a degree in this or other universities.

_____

Lee Cheng

October 2013

Thesis Examination Panel Approval

Members of the Thesis Examination Panel approve the thesis of CHENG, Lee defended on 10/07/2013.

Supervisors	**Examiners**
Prof. LEONG, Samuel Professor and Head of Department Department of Cultural and Creative Arts, Hong Kong Institute of Education	Prof. REES, Fred J. Professor Department of Music and Arts Technology, Indiana University-Purdue University Indianapolis
Prof. TSANG, Yip Fat Richard Professor Department of Cultural and Creative Arts, Hong Kong Institute of Education	Prof. NIERMAN, Glenn E. Steinhart College Professor Glenn Korff School of Music, University of Nebraska–Lincoln
Prof. SPLITTER, Laurance J. Professor Department of Cultural and Creative Arts, Hong Kong Institute of Education	Dr. LEUNG, Bo Wah Associate Professor Department of Cultural and Creative Arts, Hong Kong Institute of Education

Approved on behalf of the Thesis Examination Panel:

_____

Chair, Thesis Examination Panel
Prof LO, Sing Kai
Chair Professor
Faculty of Liberal Arts and Social Sciences, Hong Kong Institute of Education

ABSTRACT

A Model of Sustainable Ecosystem for Software Development,

Software Business and Music Education

by CHENG, Lee

The Hong Kong Institute of Education

Abstract

This interdisciplinary study addresses the issue of creating a sustainable ecosystem consisting of the ecologies of software development, software business and music education. These are three key areas contributing to the end users' applications of music software in schools. Although these areas have been developing for several decades, no existing study has been found that examines how their individual contributions combine to achieve a greater effect. An ecological approach that examines the relationships and interactions within and between each of the ecologies has also not been found. This study examines the ecological effect of software development, software business and music education, the key components and dynamics of the three ecologies and how they connect and interact with one another. The main aim of this study was to develop a model of sustainable ecosystem consisting of the ecologies of software development, software business and music education.

The overarching research question of this study was:
How can the ecologies of software development, software business and music education work together to create a mutually beneficial and sustainable ecosystem?

The research design comprised three phases using a qualitative-dominated approach. Phase I involved a review of the literature in order to map the three ecologies of software development, software business and music education. This informed the study's conceptual framework and the research approach, and identified the components of the ecologies and their relationships and interactions. A preliminary model of each of the ecologies was constructed

based on the literature review. The literature review also provided a basis for establishing the survey and interview questions for Phase II.

Phase II consisted of four separate studies, three focusing on the three ecologies and one on domain experts of technology in music education. Semi-structured interviews and a questionnaire survey were implemented with software engineering lecturers, music software developers, music software retailers, music teachers and domain experts of music technology in education. Data analysis provided insights into the dynamics within and between the three ecologies and refined the preliminary models from Phase I.

Phase III involved the development of a sustainable ecosystem for software development, software business and music education, including the key threats to each ecology. The model provides a meta-view of the ecologies of software development, software business and music education. Recommendations for achieving a sustainable ecosystem consisting of the three ecologies were provided.

Results showed that the music software development process, being the core activity of the ecosystem, requires the cooperation of music software developers, music software retailers and music teachers to share insider knowledge pertaining to their respective ecologies and contribute to interdisciplinary software development. Interdisciplinarity and knowledge transfer are the key conditions for the creation of a sustainable ecosystem.

The investigation of the three different ecologies that were usually studied in isolation contributes to current knowledge involving interdisciplinary collaborations. The revealed gaps between the ecologies and threats to the ecosystem would enhance the sustainability of getting better-designed music software more effectively delivered by music software retailers for more widespread applications in school music education.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Prof Samuel Leong, for his guidance and support. Without him, it would be impossible to complete this study. I am fortunate to have had Samuel as my supervisor who is such an experienced supervisor and mentor.

I am also indebted to Mr George Mitcheson and Prof Tse Tsun Him of the University of Hong Kong, Mr Eric Yung who is the founder and CEO of Playnote Limited, Dr Ann Blombach who is the founder of MacGAMUT Music Software International, Dr William Thorpe who is the developer of *Sing & See*, Mr. Kenny Lui from Tsang Fook Piano Limited, Mr Javan Green from Millennium Music Software Limited, Dr Andrew King of the University of Hull, Dr Evangelos Himonides of the University of London, Dr Fred J. Rees of the Indiana University – Purdue University Indianapolis, Mr Matti Ruippo of the Tampere University of Applied Sciences, and all the music teachers who participated in this research study. Their professional opinions and experiences in software engineering education, music software development, music software business, music software in education and school music education helped me in developing a sustainable ecosystem for software development, software business and music education.

I would also like to thank all the music staff and students from the Department of Cultural and Creative Arts, as well as all the hallmates from the Northcote Hall of the Hong Kong Institute of Education. Their encouragement and support make my PhD journey not a lonely one.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CAI	Computer Assisted Instruction
CEL	Computer Enhanced Learning
ICT	Information Communication Technology
IT	Information Technology
MIDI	Musical Instrument Digital Interface
MP3	Moving Picture Experts Group (MPEG) Layer 3
PGCE	Postgraduate Certificate in Education
VTF	Real-time visual feedback
UGC	University Grants Committee
UNESCO	United Nations Educational, Scientific and Cultural Organization

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Software development, software business and music education are three different areas that contribute to the eventual applications of software in school music education. While these areas have been developing for several decades, there is no evidence that they have worked together to make an aggregate contribution that is greater than their individual impact. Each of the three areas has its own ecological system, inhabited by key players who interact with one another. This study investigates how the three ecologies, which unite to form an overall ecosystem, can be made sustainable to benefit the three key areas of software development, software business and music education.

1.1. Motivation of the Study

I have long-standing interests in music, music education, computer science and business. I have been composing, performing and occasionally conducting since my years as an undergraduate, including being a band flautist for many years. As a computer science and business administration graduate, I have had the privilege of experiencing the well-developed processes of software development and software business, and have observed that these two areas rarely collaborate.

I have also been using music software for learning and making music, and my technical training has given me some understanding about deficiencies in music software design. Some software is too difficult for student-level users to use, and most software require users to possess a high level of technical competence.

As a computer science student who undertook an honours project to develop piano performance assessment software (Cheng, 2010), I have experienced how heuristic thinking can be applied to music software development. I am also interested in knowing how music software is developed in the real world, how it is marketed and how feedback is communicated to the software developers.

Through my experiences, I have come to realise the importance of interdisciplinary studies as the world becomes more complex and diversified. Interest in interdisciplinary studies in higher education has been increasing over the years (Newell, 2007). Many international research-oriented organisations have been founded, such as the International Network of Inter- and Transdisciplinarity (founded in 2010), the Philosophy of Interdisciplinarity Network (founded in 2009) and the Center for the Study of Interdisciplinarity (founded in 2008).

In this thesis, I am interested in investigating how the three ecologies of software development, software business and music education work individually and interactively, and how the dynamics within and between the ecologies may be synergised to create a sustainable ecosystem.

1.2. Background

1.2.1 Software Development

Computer software has contributed to society for more than half a century. During this time, ever-advancing computer technology has sequentially led to an increase in the quantity and functionality of software technology for business, professional and educational purposes. Software designed for different disciplines and purposes requires the application of interdisciplinary knowledge in the development process. For example, business software development requires in-depth understanding of how the business sector works and how the software could help solve particular business problems. Mathematics software development requires accurate formulation and translation of mathematical theories into computer algorithms. Software development for arts education purposes requires adequate consideration of the contexts in which software are used for learning.

Software development encompasses a range of developmental activities including planning, prototyping, coding, modification, reuse, re-engineering, maintenance and any other activity that produces a product. In the 1960s, there was an agreement that software should be developed according to a planned-out and structured process, an agreement that resulted in software engineering.

Software engineering, defined as the systematic approach to software development or 'the application of engineering to software' (Laplante, 2007), was first coined at the 1968 NATO Software Engineering Conference in Garmisch-Partinkirchen, Germany (Naru & Randell, 1968). Different software development models apply to developing software with different characteristics. Popular models include the waterfall, prototyping, spiral, evolutionary development, iterative-enhanced and rapid application development models. Some models follow a list of development phases sequentially, with one phase not starting until the end of the previous phase. Some are iterative, in which case the software is developed through repeated cycles and in smaller portions. Some are a combination of both. The model to apply depends on various factors such as the size of the project team, the nature of the software to be developed and stakeholder availability.

Despite the variety of models to apply within the software development process, the contextual processes are approximately the same, and the order and method for carrying out the activities are significant. A software

development process can be implemented in five to nine phases, depending on how it is divided and how the boundaries of each phase are defined. The phases of the software development process defined by Agarwal, Tayal and Gupta (2010) are described here as an example.

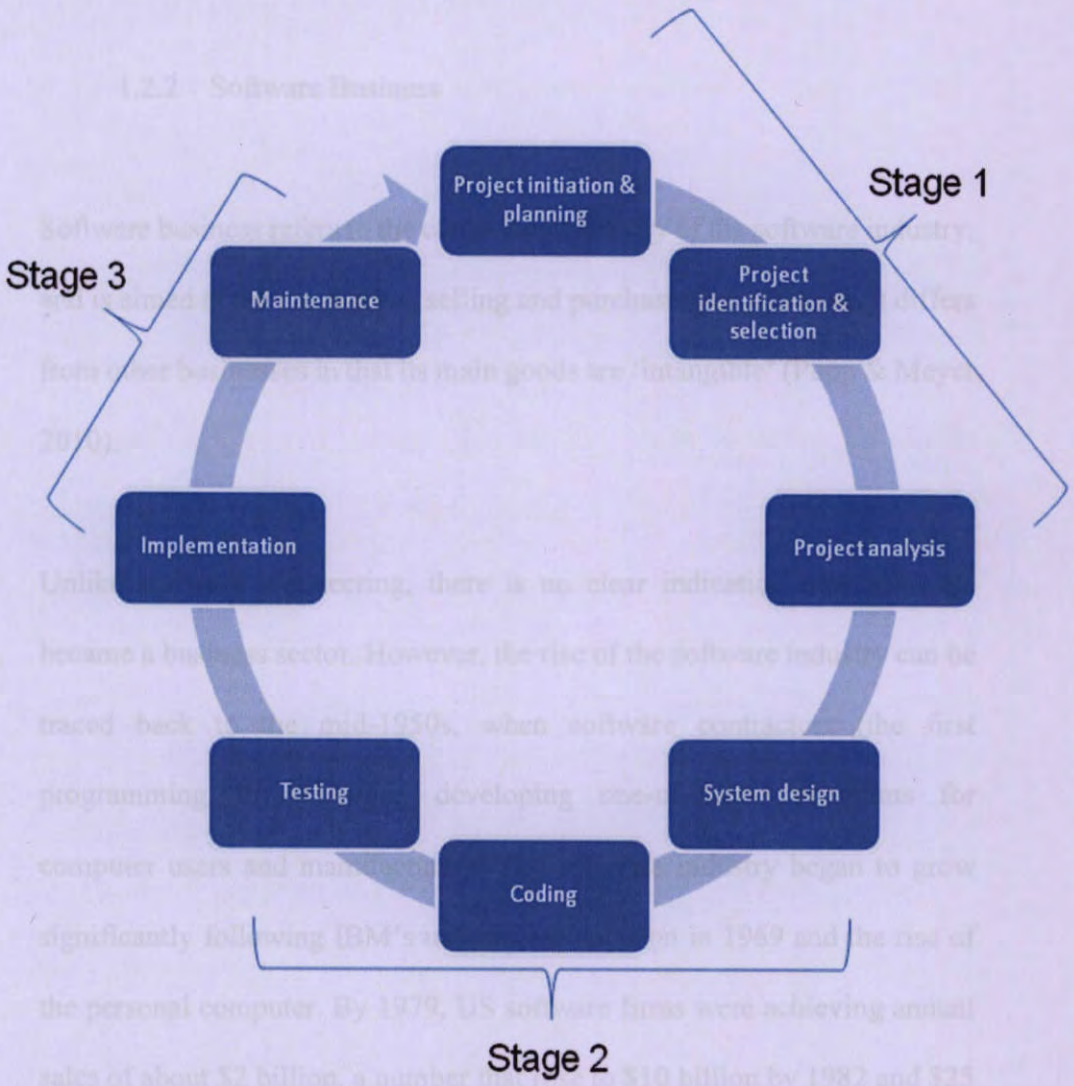

Figure 1. An example of a software development process

Activities in the software development process can be divided into three stages: planning, implementation and maintenance. The planning stage consists of the initiation of the project, framework design and requirement

analysis. The implementation stage consists of the system design, coding, testing and documentation. The maintenance stage consists of deployment, training, support and maintenance. The boundaries of each phase or stage must be defined clearly to track progress.

1.2.2 Software Business

Software business refers to the commercial activity of the software industry, and is aimed at the production, selling and purchasing of software. It differs from other businesses in that its main goods are 'intangible' (Popp & Meyer, 2010).

Unlike software engineering, there is no clear indication how software became a business sector. However, the rise of the software industry can be traced back to the mid-1950s, when software contractors (the first programming firms) began developing one-of-a-kind programs for computer users and manufacturers. The software industry began to grow significantly following IBM's unbundling decision in 1969 and the rise of the personal computer. By 1979, US software firms were achieving annual sales of about $2 billion, a number that rose to $10 billion by 1982 and $25 billion in 1985 (Campbell-Kelly, 1995). There were several reasons for the growth of the software industry in the 1970s-1980s. First, software became a key enabler of other industries. Second, software was more commonly embedded in the products and services of other industries. Third, other

industries were becoming increasingly knowledge driven and thus developed management problems similar to those of the software industry. The operations and competitiveness of e-businesses came to rely on effective software use (Hoch, Roeding, Purkert, Lindner & Mueller, 1999). The rise of the software business has sequentially introduced convenience, more choices, more function-specific software and lower prices.

Figure 2 shows the annual revenue growth of the US software industry over the last decade. At the end of the 2000s, revenue increased to about $145 billion (Miles, 2010). The recent development in software business can be attributed to the rise of software as a service (SaaS). SaaS has bloomed in recent decades due to the advancement of digital technology, mobile services and Internet use (Cusumano, 2004). It usually profits from subscription fees and the customisation of services for individual companies, as opposed to sales of a 'tangible' software product.

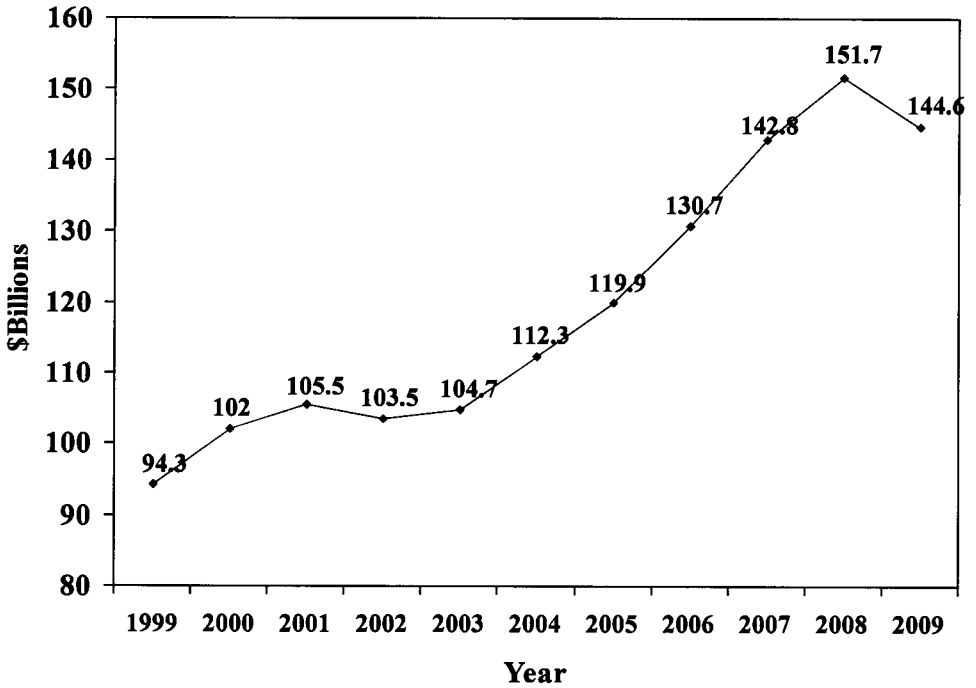

Figure 2. US software industry revenues 1999-2009 (Source: U.S. Census Bureau)

1.2.3 Music Education

Music technology particularly encompasses the use of electronic devices and computer software to facilitate the playback, recording, composition, storage, analysis and performance of music. Professionals use hardware and software such as music notation software, sequencing and mixing software, metronomes, analogue/digital recorders, music database management systems, amplification systems and electronic instruments to assist in music performance and production. The emergence of music technology has not only benefited the music industry, but also facilitated how music teachers teach and how music education is delivered.

The emergence of music technology can be traced back to the late 1930s, when inventors such as Léon Theremin and Maurice Martenot invented the first batch of electronic musical instruments, including the theremin and the ondes Martenot (Skeldon, Reid, McInally, Dougan & Fulton, 1988). In the 1970s, synthesiser technology such as Robert Moog's successful Moog synthesiser, John Chowning's invention of frequency modulation (FM) synthesis and Vogel and Ryrie's design of the Fairlight CMI marked another advancement (Chowning, 1973; Holmes, 2002; Manning, 2013).

The call for integrating technology into music education and preparing technology for music teachers can be traced back to the Tanglewood Symposium (Britton et al., 1968) held almost half a century ago in 1967, one year earlier than the NATO Software Engineering Conference in Garmisch-Partinkirchen, Germany, where the idea of software engineering was first coined. During the following decades, many changes and new initiatives have taken place in terms of applying music software to education, including online music education (Chong, 2010; Kruse, Harlos, Callahan & Herring, 2013; Montague, 2010; Schlager, 2008); the way students learn and access music (Gouzouasis, 2005; Tobias, 2012); the pedagogical approach of music teachers (Bauer, 2012; Beckstead, 2001; Savage, 2007); and the use of the World Wide Web (WWW), computer-enhanced learning (CEL), computer-assisted instruction (CAI) and information communication technology (ICT) (Breeze, 2011; Choksy,

Abramson, Gillespie, Woods & York, 2000; Ng, 2008). The advancement of technology has not only changed the way music is taught and learned, but also reconstructed the conceptual framework of music education (Cain, 2004).

Stepping into the Education 3.0[1] era, the adoption of technological and educational innovations has increased the possibilities for how teachers teach and students learn (Bauer, 2012; Leong, 2011). In response to the technological development of music education, music teachers have gradually been incorporating technology into their classroom environments. As computers have become affordable and smaller in size, music software has become more appealing and available to students. Students can easily access music notation and music-making software, musical games for educational purposes and real-time feedback software through computers at school or at home.

Music education in the 21st century has flourished with cultivated creativity, individual expression, constructive learning and other transdisciplinary learning outcomes in response to modern education needs (Leong, 2011; Partti, in press; Savage, 2005). Software technology has been used to facilitate teaching and learning, and teachers and students are able to take

[1] Education 3.0 is a term that has been used to describe a level of transformative capability and practices for education in the 21st century. It is characterised by rich, cross-institutional, cross-cultural educational opportunities within which the learners play key roles as the creators of shared knowledge artifacts. Social networking and benefits outside the immediate scope of activity also play strong roles (Keats & Schmidt, 2007).

advantage of software to create music, receive real-time visual feedback and work on music collaboratively and interactively.

While software technology has gradually been incorporated into music teaching and learning, concerns have emerged over the choice and contextual design of any music software applied to a music curriculum (Naughton, 1997). Other concerns include the fear that technology transforms music lessons from a practice of artistic cultivation to one of vocational training (Hodges, 2001), a perceived lack of consideration during the software design process and determining how the software may be integrated into music curricula (Brown, 2007; Flores, Vicari & Pimenta, 2001; Upitis, 2001).

The issues of concern highlighted above indicate the need for software developers to work closely with music teachers. If we consider software development, software business and music education as three individual ecologies (communities of living organisms that operate in conjunction with an environment's non-living components), they will need to interact more effectively for improving the complementarity of the overall ecosystem. Some of the crucial questions that need addressing include: which components of the three ecologies cater to software development to produce software that could better facilitate music education? What are the dynamics that drive the evolution of the ecologies? How do they interact with one

another? When different ecologies continue to evolve individually and rapidly, how can they be kept sustainable?

1.3. Statement of the Problem

Much research has been conducted to study the methodology, alternatives and techniques involved in software design (see Section 2.2) and to investigate the operation and success of software business (see Section 2.4). Studies have also examined the applications of music software in education and assessed its effectiveness, integration context and innovativeness of practice (see Section 2.6). While both the quantity and functionality of music software continue to grow, a dearth of research has investigated the relationships between software development, software business and music education (see Section 2.2.2).

Most of the research on such relationships has focused on particular aspects of each ecology, e.g., how music software is developed; the corresponding challenges, principles and heuristics; the lessons learned and the use of music software development as a research method (e.g., Brown, 2007; Flores et al., 2001; Krüger et al., 1999; Leong, 2002). Interdisciplinary research examining the relationships and interactions between the three ecologies has not been found. An ecological approach to examining the relationships and interactions has also been lacking. Barrett (2012) mentioned that the 'ecological approaches to education are often ignored, if

not actively discouraged' (p. 209). This applies not only to education, but also to all the three ecologies in question. There are many reasons for the lack of cross-disciplinary ecological collaboration, such as the inability of researchers to deal with problems due to their complexity and the extent of their own professional knowledge, the sector-based division of responsibilities in contemporary society and the increasingly diverse nature of social contexts (Lawrence, 2010). Ecological research has much to contribute to our understanding of music development, engagement and learning (Barrett, 2010). Welch (2010) stressed the importance of ecological impact and validity in ensuring the relevance of research and practice.

To understand the ecological effect of software development, software business and music education, the key components and dynamics of the three ecologies and how they connect and interact with one another need to be examined. A holistic understanding of the ecological effects would inform the possibilities for creating a sustainable ecosystem that consists of the three ecologies.

1.4. Main Aim

The main aim of this study was to develop a model of sustainable ecosystem consisting of the ecologies of software development, software business and music education. The relationships and interactions between the key

components in the three ecologies are examined, and a meta-view of the three ecologies forming a sustainable ecology is presented in this thesis.

Figure 3 shows my initial observations about the connections between the three ecologies of software development, software business and music education. There is an obvious connection between software development and software business as software is developed in the ecology of software development, distributed through the ecology of software business and reaches the end users who apply software in the ecology of music education. Although a connection between the ecologies of software development and music education is observed, the primary dynamics are unclear. The dynamics within and between these ecologies are also unclear. Hence the rationale for this study.

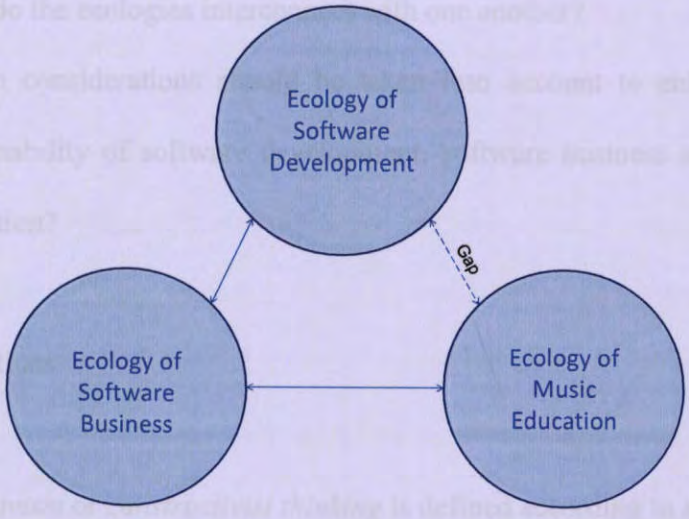

Figure 3. Gap between the ecologies of software development and music education

1.5. Research Questions

The overarching research question of this study was:

How can the ecologies of software development, software business and music education work together to create a mutually beneficial and sustainable ecosystem?

The following research questions guided the study.

1. What are the dynamics within each of the ecologies (software development, software business and music education)?
2. What are the principal dynamics of an overall ecosystem that consists of all three of these ecologies?
3. How do the ecologies interconnect with one another?
4. Which considerations should be taken into account to enhance the sustainability of software development, software business and music education?

1.6. Definitions

Constructivism or *constructivist thinking* is defined according to a complex mosaic of the beliefs of philosophers, researchers and practitioners. Webster (2011a) once summarised four aspects of constructivism as

follows. First, knowledge is formed as part of the learner's active interaction with the world. Second, knowledge does not exist as a collection of abstract entities outside of and absorbed by the learner; rather, it is constructed anew through action. Third, meaning is constructed with this knowledge. Fourth, learning is in large part a social activity. However complicated, the essence of constructivism defined here was best represented by Webster as the theory of teaching practice, which states that learners construct new understanding using what they already know and that learning is active rather than passive.

Curriculum refers to 'the content of a particular subject or area of study' (Kelly, 2009, p. 7).

Interdisciplinary approach refers to the 'inquiries which critically draw upon two or more disciplines and which lead to an integration of disciplinary insights' (Haynes, 2002, p. 17).

Knowledge transfer refers to the transferring knowledge from one part of the organisation to another. It seeks to organise, create, capture or distribute knowledge and ensure its availability for future users.

MIDI refers to the industry standard protocol that enables electronic musical instruments, computers and electronic equipment to communicate with one another.

Music notation software refers to computer programs that are used to create sheet music, also known as score writers.

Pedagogy refers to 'the study of the methods and activities of teaching' (Cambridge Dictionary Online, 2012). The use of this term as an

adjective, e.g., pedagogical skills and pedagogical considerations, describes the object as teaching-method-oriented and activities-oriented.

Requirement engineering refers to 'the systemic use of proven principles, techniques, languages, and tools for the cost-effective analysis, documentation, and ongoing evolution of user needs and the specification of the external behavior of a system to satisfy those user needs' (Marciniak, 1994, p. 1177).

Software requirement analysis refers to 'the process of classifying requirements information into various categories, evaluating requirements for desirable qualities, representing requirements in different forms, deriving detailed requirements from high-level requirements, negotiating priorities, and so on' (Wiegers, 2003, p. 483).

1.6.1 Ecology

The term 'ecology' was originally a Greek word ('οικολογία') meaning the study of a dwelling, such as a planet. According to its popular scientific meaning, ecology is the study of the relationships that living organisms have with one another and with the natural environment. It is always associated with another term, 'ecosystem', which is defined as a community of living organisms existing in conjunction with an environment's non-living components and interacting as a system (Smith & Smith, 2012). If we

believe that our living environment may be organised into a system, we could define ecology as the study of an ecosystem.

The term 'ecology' has long been used as a metaphor in the business sector due to the similarity of the interactions between organisations and individuals with the living organisms of an ecosystem. James F. Moore originated the term 'business ecology' in the early 1990s. In his book entitled *The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems* (1996), Moore defined the term as follows:

> *An economic community supported by a foundation of interacting organizations and individuals—the organisms of the business world. The economic community produces goods and services of value to customers, who are themselves members of the ecosystem. The member organisms also include suppliers, lead producers, competitors, and other stakeholders. Over time, they coevolve their capabilities and roles, and tend to align themselves with the directions set by one or more central companies. Those companies holding leadership roles may change over time, but the function of ecosystem leader is valued by the community because it enables members to move toward shared visions to align their investments, and to find mutually supportive roles.* (p. 26)

Ecology has been used as a metaphor for the education field, as it describes the practice of educational communities and serves as a model for educational policy making. For example, Goodlad (1985) summarised and described a healthy educational ecology as 'one in which each institution fulfills clear and appropriate functions, none is overburdened, and there is much collaboration in alternative formal, informal, and nonformal ways of educating children and youth' (p. 43).

The ecology and ecosystem metaphors applied to music education and business practices share some characteristics. They both rely on systematic constructions to guide their methods, and in each ecosystem there are dynamics that interact with one another to make the system work. The term 'dynamics' is introduced in the following section.

1.6.2 Dynamics

'Dynamics' is another Greek word ('δυναμικός' or 'δύναμις') meaning 'power' or 'powerful'. It is mainly used in the field of physics, which examines causes of and changes in motion. Similar to the term 'ecology', 'dynamics' has been used as a metaphor in other fields such as sociology, psychology, music and computer science.

In this thesis, 'dynamics' is defined as both the group behaviour resulting from the interactions of individual group members and the study of the

relationships between individual interactions (Durlauf & Young, 2001). Individuals are influenced by one another's behaviour. Their interactions are unidirectional and change the ecology as a whole.

1.6.3 Sustainability

'Sustainability' refers to the capacity to endure. It comes from the original Latin word 'sustinere', which means both 'to hold' ('tenere') and 'up' ('sus'). From an ecological perspective, the word describes how biological systems remain diverse and productive over time. In this study, sustainability means all threats in the ecosystem are removed, and all the dynamics work within and between each of the ecologies.

This study examined the dynamics that contribute to the sustainability of the ecosystem aforementioned. The three ecologies interacted with each other to maintain ecological soundness. In addition, the ecologies interconnected to create a larger ecosystem.

1.7. Significance of the Study

By investigating three different areas that are usually studied in isolation, this study contributes to the literature by clarifying the key components of the three ecologies and identifying the dynamics within and between the ecologies using an interdisciplinary approach. The findings are expected to

reveal any gaps between the ecologies and provide a more holistic understanding of the overall ecosystem to make it more sustainable. Music software could benefit from having a better development methodology, software business could benefit from having an enhanced functional role to bridge the disconnection between software developers and music teachers, and the latter could benefit from having access to music software that is appropriate for use in school music education.

1.8. Delimitations of the Study

Music education in this study was confined to the high school and higher education sectors. This study was delimited to the applications of commercially produced software used in music education. General computer software such as word processing software, spreadsheet software, database management software, Web browsers, media players and information sharing platforms were excluded, as were free and open-source music education software. Only software that included English and Chinese language options was examined.

The scope of the sustainable model to be developed was bound by the primary activities that characterised the three ecologies of software development, software business and music education. While some elements outside this scope may be found, only the primary activities and critical interactions would be included.

1.9 Organisation of the Thesis

This chapter has provided an overview of the study's background, motivation, problem statement, aims, research questions, abbreviations, definitions, significance and limitations.

The rest of the study is organised into eight chapters. Chapter 2 serves as a review of literature pertaining to relevant research findings on the three ecologies to gain preliminary understanding of each ecology. Chapter 3 details the methodology, including the epistemological foundations on which the research was conducted. Chapters 4-6 describe and discuss the findings pertaining to the three studies involving key players from the three ecologies as participants. Chapter 7 presents and discusses the findings from domain experts of music technology in education. Chapter 8 describes the model development of a sustainable ecosystem consisting of the ecologies of software development, software business and music education, and recommends ways to address the threats to the ecosystem.

CHAPTER 2

Literature Review

This literature review covers research on the key components of the three ecologies investigated in this study, including (1) software engineering education; (2) software development; (3) software business; (4) music software in school education; and (5) music technology in higher education. The literature on ecological sustainability was also reviewed.

The sections in this chapter follow an operational flow, from how music software is developed to its applications in music education, as illustrated in Figure 4. Software developers are systematically trained in software engineering, and apply their knowledge to music software development. The developed software is then distributed through a software business and reaches the music education sector, where music teachers apply it to school music education. The software is also applied in higher music education and teacher education programmes, the latter of which prepares music teachers to apply music software to school music education.

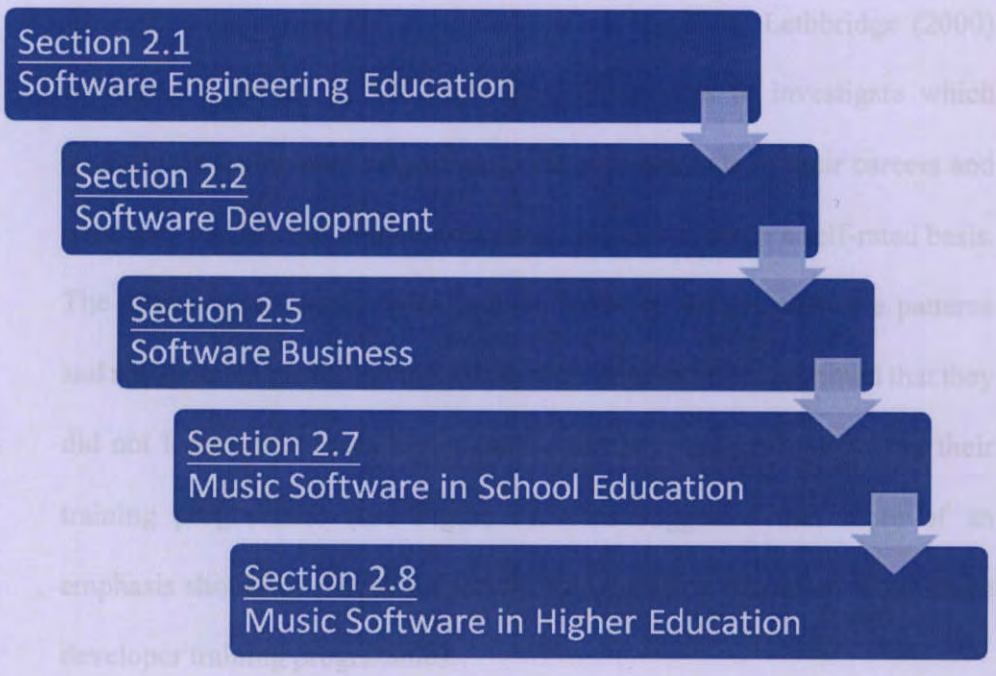

Figure 4. Organisation of the literature review

2.1 Software Engineering Education

Unless they are self-trained, most software developers acquire their skills and knowledge from computer science tertiary education. While technical skills related to computer architecture and coding are essential to real-world software development, software engineering has been considered crucial to the quality of software products (van Vliet, 2008). It involves the application of a systematic, disciplined, quantifiable approach to the design, development, operation and maintenance of software (Abran, Moore, Bourque, Dupuis & Tripp, 2004).

Aiming at improving the computer science curricula, Lethbridge (2000) conducted a survey of 186 software professionals to investigate which educational topics were important to the professionals in their careers and what they had learned in their education and on the job on a self-rated basis. The professionals rated topics such as software design, software patterns and requirement gathering/analysis as very important, but admitted that they did not learn about these topics until after they had graduated from their training programmes (see Figure 5). This suggested that more of an emphasis should be placed on software engineering education in software developer training programmes.

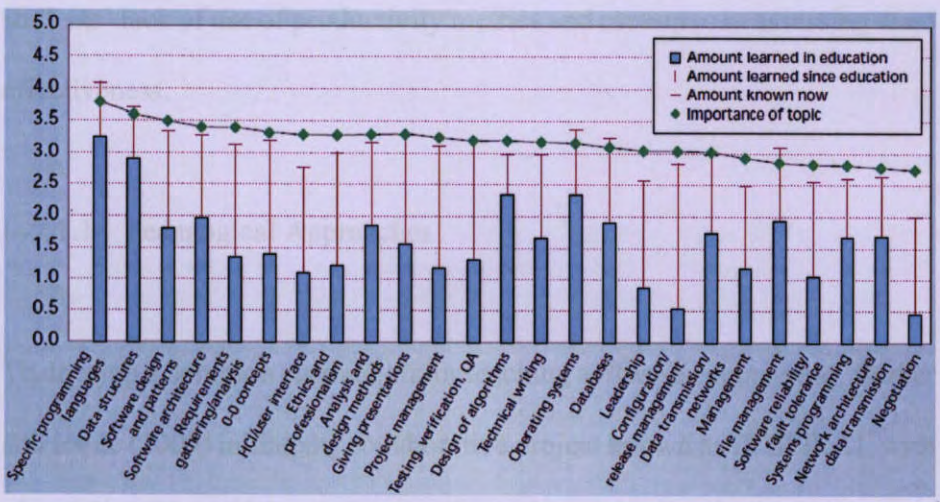

Figure 5. Twenty-five most important topics rated by software professionals

With an insight to remove threats in the practicum component of software engineering curricula, Bareiss and Katz (2011) conducted a survey of graduate-level software engineering students who completed a team-based practicum to ascertain their most significant self-reported shortcomings.

The authors also presented previous practicum advisors' observations on the problems the students experienced with the knowledge transfer from the formal curriculum to the relatively unstructured practicum project environment. The advisors noted issues such as poor testing or no test suites, too much concern over satisfying clients, ineffective teams, misperception of particular software development cycles, the absence of productivity metrics and the absence of clear project quality measures in the students' practicums. According to the survey results, these issues were caused by the clients' lack of experience with software engineering principles and procedures and the students' over-reliance on client feedback as the predominant mechanism for quality assurance. The survey also revealed the students' lack of use of productivity metrics and measures in assessing team effectiveness.

2.1.1 Pedagogical Approaches

To develop a common curriculum for teaching software engineering, Barker and Inoue (2009) initiated a collaborative project known as IT SPIRAL with nine universities and four industries. It combined the foundational education practices at the individual universities, a shared DVD library on advanced software engineering topics and common intensive sessions led by industry participants. It showed that cooperation between universities and industry could provide a wider range of educational materials and more depth, and

that the direct involvement of the industry in the educational experience benefited the universities, students and industry alike.

Gannod, Burge and Helmick (2008) treated other uses of digital media such as video as formal lectures in their inverted classroom approach to software engineering education. An inverted classroom is a teaching environment that mixes technology with hands-on activities. The authors replaced typical in-class lecture time with laboratory and in-class activities. The students' response was positive, especially their response to the inverted classroom model. In terms of replacing lecture hours with in-class activities, Wu, Chen, Wang and Su (2008) developed and evaluated a game-based learning system for a software engineering course. In the game-based learning environment, learners played different characters such as project leaders, system analysts, system designers and programmers, and conducted collaborative activities with other team members during the software development process. The results of a questionnaire administered to 34 undergraduate course participants indicated that the students had a positive learning attitude toward using the system.

The problem-based learning (PBL) approach has been adopted recently in software engineering education. For example, Santos, Batista, Cavalcanti, Albuquerque and Meira (2009) proposed an innovative pedagogical methodology based on PBL to improve learning effectiveness in a graduate level software engineering course. The authors received a positive

evaluation from the software companies that the students worked with. Brodie, Zhou and Gibbons (2008) investigated the use of PBL in higher software engineering education programmes, in which they observed students benefiting from foundational skills such as multidisciplinary teamwork, solving complex problems and communication. They also revealed the significance of using real work problems. However, the implementation of PBL in software engineering education requires new skills for the lecturer to transform himself or herself into a facilitator. More time and emphasis must be placed on incorporating required learning objectives into problem solving.

Collaboration is key to PBL in software engineering education, as students work together to conduct meetings, make group decisions, manage groups, write reports and deliver presentations. Anisetty and Young (2011) identified several specific tools for the different phases of the software development process that may enable students to work better collaboratively in enhancing their performance. They found different tools such as Google Docs, Subversion, Visual Paradigm and Netbeans to be useful in helping students improve their collaborations.

Case studies are often used in contemporary software engineering education and offer a similar approach to PBL. In addition to PBL, students are provided with the background of the people and organisations involved, the nature of their work, the challenges they face and the decisions they make,

which requires an analysis of their problems; applications of concepts, tools, techniques and skills; evaluations of alternatives and decisions. Many successful examples (e.g., Gary & Varma, 2007; Hilburn, Towhidnejad, Nangia & Shen, 2006) have shown the advantages of the case study approach over the traditional lecture-based approach, such as its bridging of the industry-academia gap, its creation of professionals who are well versed in theory and practice and its encouragement of students' active and centric engagement in the learning process.

In addition to the PBL and case study approaches, the iterative approach has been adopted in software engineering education. Sebern (1997) incorporated the iterative approach in a software engineering course more than 10 years ago. However, the students did not typically have an opportunity to apply the lessons they learned to real-world projects (*ibid.*). More recently, Gast (2008) successfully used an iterative approach in a master's-level full-year course. The students in the course became more confident about the expected outcomes as they completed each iteration (*ibid.*). Noble, Marshall, Marshall and Biddle (2004) integrated extreme programming, a kind of iterative approach advocating frequent releases in short development cycles, in a document-centric software project course. Students and clients had positive things to say about the approach. The students were able to produce working code, which was not previously possible, and clients found the code produced to be more aligned with their needs. Rico and Sayani (2009) introduced agile software development,

another kind of iterative approach, to a master's-level software engineering course, where requirements and solutions evolved through collaboration between self-organising and cross-functional teams.

The iterative approach had some drawbacks, mainly due to the time constraints of the single-semester course. First, students were reluctant to produce documentation and wait until the end of a project. Second, the instructors had increased workloads that involved addressing missing client issues, the provision of continuous feedback and making sure the projects selected were development ready and the ideas concrete in nature. Third, the time students had to finish iterations between changes in group composition was insufficient (Khmelevsky, 2009; Kokkoniemi, 2008; Sebern, 1997).

The concept of dynamic group composition was introduced by Schwartz (2008), who observed that student group composition should rotate at regular intervals to improve the quality of the documentation created by the groups. Anewalt and Polack-Wahl (2010) applied dynamic group composition to their software engineering course and found that there were fewer group breakdowns when team members underperformed. The students benefited from the practice of reading other people's work, adapting to multiple personalities and skillsets and being able to rate peers honestly. Moreover, the course as a whole benefited from an increased project completion rate, from 30% to 80%.

Schön (1983, 1987) introduced the reflective practitioner (RP) perspective, which guides professional people to rethink their professional creations during and after the accomplishment of the creation process. The RP perspective emphasises the studio as the education environment, in which students develop design projects under the close guidance of a tutor. Many examples have shown the effectiveness of university software studios and studio-oriented university programmes (cf. Tomayko, 1991, 1996; Kuhn, 1998; Docherty, Sutton & Brereton, 2001; Docherty et al., 2001). Hazzan (2002) examined the fitness of the RP perspective to software engineering education based on visits to studios and conversations with tutors and practitioners, which reinforced the importance of reflective proficiency to software engineering education.

2.1.2 Curriculum Updates

Updating curriculum content is an important issue in software engineering education. Before the millennium, the software engineering education sector focused on creating curricula that prepared engineers 'to work with new science and technology throughout their careers and helping them become agents of change in the industry' rather than training undergraduates to solve current problems with current technology (Garlan, Gluch & Tomayko, 1997). A traditional software engineering education that focuses heavily on software engineering methodologies is simply not adequate, as it emphasises only up-front planning and follows a linear path

through development (Hogan, Smith & Thomas, 2005; Parnas, 1999). Shaw (2000) made the following prediction:

> *Over the next decade, education for software developers should prepare students differently for different roles, infuse a stronger engineering attitude in curricula, help students stay current in the face of rapid change, and establish credentials that accurately reflect ability.*

In addition to continuing traditional computer science education, the changing landscape of pervasive computing requires new software engineering teaching foci, such as how to develop safety-critical systems and secure software and how to shorten transition times for students from classroom to industry. Huang and Distante (2006) suggested that software engineering instructors should motivate students' interests and maintain their enthusiasm, and this requires a new pedagogical approach. Deploying real-world professional software development tools in students' assignments could minimise the gap between academic education and industry demands, and provide an incentive for students to complete high-quality work and attain the sense of fulfilment that accompanies making a difference in society.

There are precautions to be taken when the software engineering community gradually updates and transforms its curricula. Van Vliet (2006) identified five assumptions that could trap software engineering educators: (1) that software engineering courses must include an industrial project; (2)

that software engineering is a stereotypical branch of engineering; (3) that software engineering planning is improper relative to other fields; (4) that the user interface is less important than it actually is and (5) that the Guide to the Software Engineering Body of Knowledge (SWEBOK) represents the state of the software engineering practice. These assumptions resulted from student overloading, an ignorance of human factors, the lagging behind of software engineering authorities and the application of the findings of infrastructure projects in other fields directly to software engineering curricula. Vliet suggested that software engineering practitioners should reconcile the engineering dimension with the human and social dimensions, not just software engineering *per se*.

2.1.3 Observations

The review of literature on software engineering education have suggested that the discipline is shifting from a traditional lecture-based model to more liberal and dynamic pedagogies, such as the use of an iterative approach, case studies, PBL, teamwork and industrial involvement. These pedagogies have addressed human factors in software engineering education theory and practice, resulting in an increased motivation and student-centred learning and bridging the gap between the software engineering industry and academia. From an educational perspective, these more dynamic approaches have also prepared students to face rapid changes and

challenges in the software engineering industry. In the following section, the literature review looks into how music software has developed.

2.2 Software Development

While a large amount of literature has examined how generic software has developed, very few studies have focused particularly on music software development. The research on software technology and music education has focused on the application and integration of technology in music education, rather than how the technology is designed and modified.

Among the several major processes in software development, requirement analysis is particularly important to music software development. While the other developmental processes involved in music software are not that different from those of any general software, requirement analysis defines the functions of the software in addition to the user interface and other non-functional attributes.

This subsection looks into the research and practice of requirement analysis before considering the literature on music software development.

2.2.1 Requirement Analysis

Requirement analysis is often recognised as one of the most critical activities in the software development cycle (Bourque, Dupuis, Abran, Moore & Tripp, 2004). Much of the research has pointed out the importance of requirement analysis in determining the quality of any software being developed.

The Standish Group (1994) found that 74% of surveyed software development projects fail, mainly due to requirement problems such as a lack of user input and clear statement of requirements. Jones (1996) discovered requirement engineering to be deficient in more than 75% of enterprises. Given those unexpectedly high percentages of software failure, much research has looked into the requirement analysis process. Rajagopa, Lee, Ahlswede, Chiang and Karolak (2005) found that 70% of system errors occurred due to inadequate requirement gathering and analysis.

To investigate the state of practice of software engineering in software development, Neill and Laplante (2003) conducted an empirical study to survey 194 software development practitioners including system designers, project managers and technical specialists in various software industries and universities. Their key findings showed that 33% of the participants did not use any methodology for requirement analysis and modelling, meaning that formal requirement analysis models were rarely used. Over 50% of the participants who did use such a model applied scenarios or cases for requirement elicitation, representation and modelling purposes. In contrast,

only 30% of the participants used object-oriented analysis, which was often applied along with use cases.

2.2.1.1 Key Success Factors and Good Practices

To investigate the key success factors of requirement engineering, Hofmann and Lehner (2001) surveyed and interviewed 15 requirement engineering teams to elicit the key success factors of their processes. These factors included an in-depth knowledge of the application domain, IT, the requirement engineering process, stakeholder feedback, identification of the boundaries of the application domain and stakeholders, the resources invested in requirement engineering and stakeholders' requirement prioritisation.

Sommerville and Sawyer (1997) proposed 66 practices that would lead to improving the requirement engineering process and ultimately business benefits. The practices were divided into eight major key areas, including requirement documentation, requirement elicitation, requirement analysis and negotiation, requirement description, system modelling, requirement validation, requirement management and requirements for critical systems. A decade later, Cox, Niazi and Verner (2009) examined the first seven key practice areas by conducting in-depth interviews with 10 Australian software development organisations to single out the most standardised or valuable practice. The results showed that making business cases; assessing

system feasibility; defining system boundaries; specifying requirements quantitatively and defining standard requirement templates; using a data dictionary; proposing test cases and defining change management processes were the most popular practices corresponding to the seven key areas mentioned.

2.2.1.2 Agile Software Development

To investigate the state of practice of agile software development, Cao and Ramesh (2008) conducted an empirical study of 16 software development organisations that used agile approaches for requirement engineering. The study identified seven agile requirement engineering practices to be investigated in the participating organisations, including face-to-face communication over written specifications; iterative requirement engineering; extreme requirement prioritisation; managing requirement changes through constant planning; prototyping; test-driven development and the use of review meetings and acceptance tests. The results showed that the most common challenges were the inability to gain access to the customer and obtaining a consensus among various customer groups. Test-driven development was the least-used practice, as most developers were not accustomed to the discipline it required. One third of the participating organisations did not practice prototyping despite its popularity. The study concluded that agile requirement was more dynamic and adaptive than other established requirement engineering practices,

which may suggest why agile programming has grown in popularity in recent years.

Maiden and Jones (2010) took the emergence of agile requirements as a starting point to argue that software developers must rethink how to document user requirements. While agile requirements emerge to encourage communication and collaboration with end users and develop software without the need for modelling, they represent a counter response to traditional document-led software development practices. However, there has been little support for discovering and analysing business goals, describing and reasoning out non-functional requirements or developing more innovative ideas due to short-term development sprints. New technologies such as media-rich communication tools have been expected to offer new opportunities to agile requirements.

The aforementioned review revealed several characteristics of the state of practice in the requirement analysis process. First, not much emphasis has been placed on requirement analysis in terms of the whole software development process, which could lead to software failure. Second, there have been several approaches to requirement analysis, which have varied from the nature of the software development company/team to the resulting outcomes. Third, key success factors and good practices have been identified. The results of the literature review in this subsection suggest that the requirement analysis process requires more effort, including but not

limited to the selection of an appropriate requirement analysis approach, a focus on the key success factors and the adoption of best practices.

2.2.1.3 Innovative approaches

Robertson (2004) introduced the approach of using a business case as the project driver to perform requirement analysis. Business cases provide mechanisms for justifying and guiding software production projects, thereby helping to decide the most profitable investment of effort. The SGS (scope, goals, stakeholders) model shown in Figure 6 involves gathering relevant information about the business case to design user requirements. The business case approach benefits the requirement analysis process by driving out misunderstanding and uncertainty, and the change in requirements can be traced through the feedback loop between the stakeholders, requirement analysts and designers.

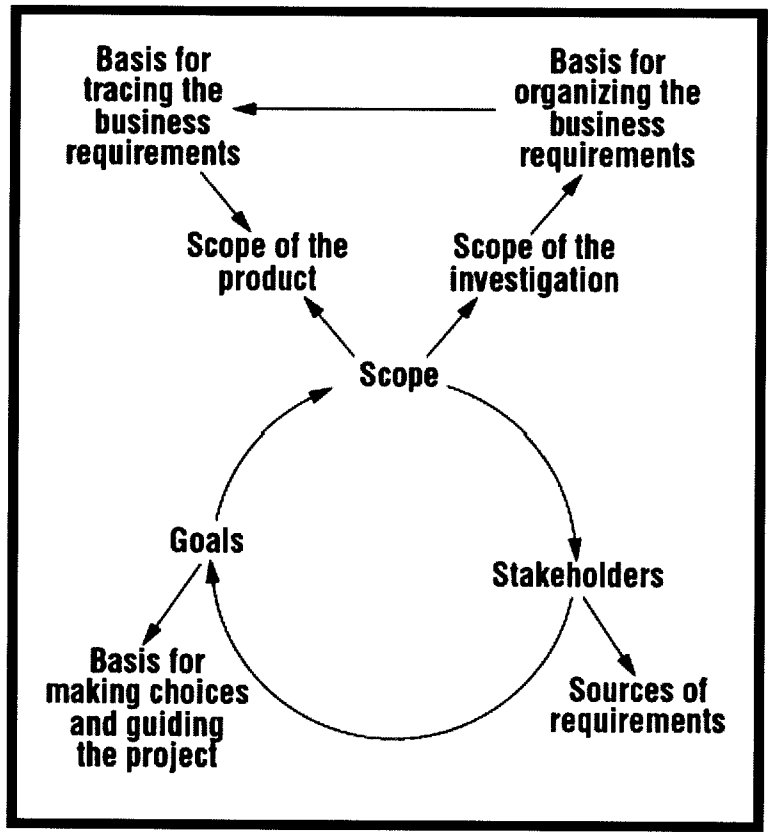

Figure 6. SGS model for the business case approach

Using a similar approach to that of Robertson, Favaro (2002) discussed three ways to maximise business value by investing in requirement engineering, including (1) actively managing the requirements process to replace the traditional passive approach, where the requirements arrive from stakeholders; (2) implementing an agile requirement process in which the iterative development paradigm breaks the tradition of up-front requirement elicitation and analysis, allowing requirements to be introduced, modified or removed in successive iterations and (3) allowing contractual flexibility for software developers to delay, select, modify, add or even abandon the implementation of requirements for which utility is uncertain.

Decker, Ras, Rech, Jaubert and Rieth (2007) researched wiki-based stakeholder participation in requirement engineering. The wiki approach to requirement engineering provides a flexible platform for asynchronous collaboration to create content in general (Louridas, 2006). As a platform that can accommodate participants in distributed environments, the platform must account for the challenges related to stakeholders, such as different objectives, different abilities to express requirements, different extents of involvement and different perspectives on the software. In their experiment involving the implementation of wiki-based stakeholder participation in requirement engineering, Decker and his team discovered several problems such as missing replications of wiki content and versioning across different pages.

Through a case study, Sutherland and Maiden (2010) investigated the use of storyboards to extract software requirements from stakeholders. Storyboarding may capture the interplay between human interaction and service design, improving the quality of service design delivery. The advantages of using a storyboard are that it can dynamically explore requirements and help prioritise them by bringing scenarios together. By visualising the scenarios of users' needs, the engagement of stakeholders is also increased.

Ranjan and Misra (2009) proposed an agent-based open and adaptive system development process that continuously changes and evolves to meet new requirements. By separating the requirement gathering and analysis phases, the scalability of the system is increased and stakeholders find it easier to understand the problem domain.

Provided that traditional functional requirements specify the system's role but ignore the system's context, Lauesen (2003) proposed replacing the Unified Modeling Language (UML) with the task and support method in software requirement analyses. By using annotated task descriptions, the computer and user could accomplish a task together without knowing which actor performed which parts of the task. This method processes higher-quality requirements that are quicker to produce and easier to verify and validate.

To make requirement negotiations more effective, Fricker, Gorschek, Byman and Schmidle (2010) proposed the use of handshaking techniques to communicate requirements. In the proposed handshaking process, the customer would first tailor the requirements to the software developer, and then the developer would share the intended design by communicating implementation proposals to the customer. The customer and developer would negotiate the implementation proposal until they found common ground for the requirements and design. After this handshaking negotiation, the requirements would be documented for further implementation. This

requirement gathering approach could promote win-win negotiations between stakeholders and developers.

To cater to diverse and even conflicting needs and viewpoints from different stakeholders, Gottesdiener (2003) held collaborative workshops that gathered every stakeholder together to define their requirements in an efficient, controlled and dynamic setting. This helped them to elicit, prioritise and agree on a set of high-quality project requirements (Gottesdiener, 2002). Gottesdiener pointed out that obtaining the right participants was the key success factor, and that other minor factors such as having a good facilitator could also help in the collaborative workshop.

Maiden (2006) described two techniques to quantify requirements: 'Volere' (Robertson & Robertson, 1999) and 'Planguage' (Gilb, 2005). Volere unitises requirement types to guide the development of the fit criteria, and Planguage is a keyword-driven language that guides analysts to write measurable and testable quality requirements. To further facilitate the quantification of requirements, Maiden (2007) proposed the use of scenarios to make requirements easier to measure. This simple model, shown in Figure 7, enables the contextual details of requirements to be described through storytelling, making the requirements easier to read and simpler to quantify.

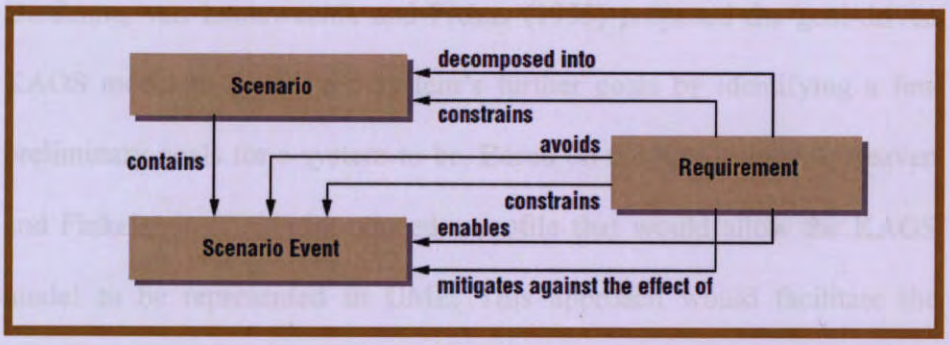

Figure 7. Simple model linking scenarios, scenario events and requirements

Stone and Sawyer (2006) proposed a tool called 'Prospect' that would retrospectively identify pre-requirement traces. The tool would split up requirement specifications and the source material from which they were derived into chunks and compare their semantic similarities to determine the probable sources of each chunk. By identifying requirements that are not firmly derived from the supplied source material, instances of tacit knowledge embedded in the problem domain would also be identified for further requirement processing.

Rapanotti, Hall and Li (2006) proposed problem reduction as an approach to derive specifications from requirements in the context of problem-oriented analysis. Through the systemic application of a small set of rules, the requirements would be decreased to specifications. This approach would allow the context of a problem to be simplified while the requirement is re-expressed.

Dardenne, van Lamsweerde and Fickas (1993) proposed the goal-driven KAOS model to facilitate a system's further goals by identifying a few preliminary goals for a system-to-be. Based on the KAOS model, Heaven and Finkelstein (2004) introduced a profile that would allow the KAOS model to be represented in UML. This approach would facilitate the unification of the KAOS requirement engineering activities with other UML design documentation, making the UML specification comprehensive and more manageable.

Non-functional requirements have frequently been neglected in software development. Cysneiros and Leite (2004) addressed this issue by treating non-functional requirements as first class requirements, presented a strategy for tackling the problem of non-functional requirement elicitation and proposed a systematic process to assure that the conceptual models would satisfy the non-functional requirements. Through three case studies, the authors suggested that the proposed approach could improve the quality of the resulting conceptual models.

With an aim to decrease software project delays, Ebert (2006) shared several key insights from his case study: (1) requirement engineering must start early and connect portfolio considerations with project management; (2) every relevant stakeholder must be available and empowered; (3) the product life cycle must be mandatory for all projects, independent of type, size and scope and (4) every element of the portfolio must be equally and

easily visible. Ebert then translated these initial observations into four requirement engineering techniques and evaluated their effect on decreasing project delays. The four techniques were (1) to install an effective core team for each product release; (2) to focus on gatekeeping reviews of the product life cycle; (3) to evaluate the requirements from various perspectives; and (4) to ensure dependable portfolio visibility and release implementation.

2.2.2 Software Development in Music Education

Few studies have focused on music software development, which requires an interdisciplinary understanding. The purpose of this subsection is to review the research undertaken on music software development, including the interdisciplinary research relevant to music education in software development and vice versa.

Music technology began to draw the attention of the music education sector when electronic music became viable in the 1950s through the development of the Columbia-Princeton Music Center in New York City. In the late 1960s, music colleges in the US began offering electronic music courses that taught musical composition, theory and techniques. During the 1970s, many schools set up electronic music studios with minimal equipment and the occasional sophisticated synthesiser and sound manipulation device. In the 1980s, the increasing portability, versatility and lower price of synthesisers made their use in live performance more feasible, and an

increasing numbers of school began using synthesisers for that purpose (Mark, 1986).

The use of computers in music education accompanied the introduction of computers in general education beginning in the late 1950s. CAI software such as PLATO (Programmed Logic for Automated Teaching Operations) was developed on a Control Data computer to offer coursework in the 1960s (Bushnell, 1962). Peters (1979), the owner of Electronic Courseware Systems, discussed the transfer of courseware from mainframe computers (PLATO) to microcomputers. David Williams, David Shrader, Sharon Lohse Kunitz, Brian Moore and other developers have developed a range of Apple II music software which provided the basis of what became the Micro Music Software library published by Temporal Acuity Products (Williams & Shrader, 1980).

The use of music software in education began to take off between the late 1960s and 1980s, whereas the National Consortium for Computer-Based Instruction was created (which has been transmuted into the Association for Technology in Music Instruction today) in 1975. Aural training software such as GUIDO (Graded Units for Interactive Dictation Operations) and MEDICI (Melodic Dictation Computerized Instruction) were developed between the decades that saw the CAI enter into music education (Hofstetter, 1978, 1979, 1980, 1981; Newcomb, Weage & Spencer, 1981). Research on the effectiveness of aural training software – such as *TAP Master* developed

in 1974 by David Shrader, who was one of theApple II music software developers – was undertaken during this period (Parker, 1979).

Williams and Webster (2008) divided the history of music and computer technology into five periods. Highlights from their summaries include the following. During the 1600s to mid-1800s, the invention of music machines depended on imaginative applications of mechanical gears such as music boxes, calliopes, mechanical organs and player pianos. During the mid-1800s to early-1900s, another batch of music machine inventions such as the Hammond organ, the Singing Arc and the Telharmonium took advantage of electricity and electric motors. During the early 1900s to mid-1900s, music machines such as Lee de Forest's oscillators and amplifiers, the theremin, the ondes Martenot and the early synthesisers of Givelet and Coupeux benefited from the unique technology provided by vacuum tubes and electromagnetic relay switches. During the mid-1900s to 1970s, music machines were transformed by transistors and semiconductors, which led to the first portable commercial music synthesisers such as the AERP and Putney. Finally, music machines became smaller and more powerful thanks to the mass production of integrated circuits, and the emergence of minicomputers and applications such as MIDI and FM.

Based on their experience of creating the Rhythmic Training System, Krüger et al. (1999) concluded that music education software development required (1) a foundation of research on music education and correlated

areas; (2) computer science resources and special techniques; and (3) interdisciplinary teams to reach the proposed aims. They pointed out that interactions between team members from different backgrounds would increase the efficiency and quality of the development process.

During their 5-year examination of music education software development, Flores et al. (2001) summarised some of the heuristics that constituted the specific development methodology for music education software. They discovered that the development of educational software rarely followed any software engineering methodology. While music software development is an interdisciplinary task that should involve expertise from music, education and computer science, they found that in most cases the musicians and music teachers did not have contact with the computer science community during the software development process. In their search for a specific development methodology for music education software, they found it difficult to achieve any general solutions because the development procedures were very user, task and context dependent.

In one music education software development project, Leong (2002) brought together a research team of teacher-educators, software engineers and music students. He identified several of the key challenges and significant lessons learned from the experiences of interdisciplinary collaboration on design and development. First, a music education software development team should be composed of personnel with expertise in one

field and at least rudimentary knowledge of the other. Second, attainable sub-goals motivate the team and keep the development process on track. Third, development tools should be chosen carefully to permit the development of every feature required by the end product. Fourth, non-interventionist and interrogatory approaches are necessary to elicit user comments. Finally, unexpected user responses in the software testing process must be addressed.

Lenberg (2010) interviewed 28 users, domain experts and stakeholders in music software produced by Propellerhead Software (which produces the digital audio workshop known as *Reason* as its flagship product) to learn more about why people want to make music and why some users find music applications difficult to learn. He also investigated the company's design process and how it affected the company's products. The findings showed that music software companies are not implementing user research as a foundation for product design. Rather, they are making technology-focused applications with an emphasis on new features. Their main goal is to satisfy existing users, and they make little effort to target a broader audience beyond current user types. This is problematic, as people generally want to make music for the fun of it, but for many new users the technical barrier is too high.

Although the aforementioned literature has focused mainly on how music education software is developed in addition to the corresponding challenges,

principles, heuristics and lessons learned, Brown (2007) asserted that software development could be used as a music education research method. He presented a new approach to arts education research known as 'software development as research' (SoDaR), which consists of three stages: (1) identification of the learning opportunity; (2) design and production of the software; and (3) implementation and refinement of the software via application in an educational setting. This approach enables new ideas about interaction, understanding or behaviour to be tested through activities using purpose-designed software that facilitates specific interactions.

2.2.3 Observations

The literature review revealed that current music software development practices not only lack the good formal practices of requirement analysis and appropriate teamwork, but also the interdisciplinary approaches that ensure their success. This interdisciplinary area requires stronger collaboration between those with expertise in the two ecologies of software engineering and music education.

2.3 Ecology of Software Development

The literature review has provided a preliminary understanding of the ecology of music software development, which is shown in Figure 8.

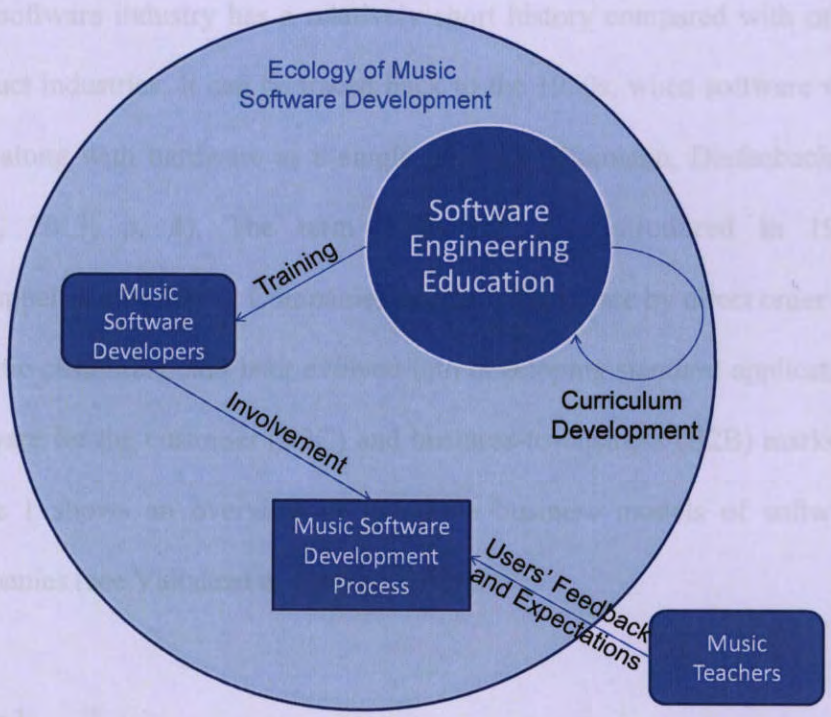

Figure 8. Ecology of music software development

Figure 8 shows that music software developers receive their training from educational institutions which should continually update and develop their curricula. Music software developers should consult music teachers during the requirement analysis process and take heed of users' feedback and expectations. This model also shows that the software business should act as a bridge between software developers and the end users who purchase the software for music education purposes.

2.4 Software Business

The software industry has a relatively short history compared with other product industries. It can be traced back to the 1950s, when software was sold along with hardware as a single package (Buxmann, Diefenbach & Hess, 2013, p. 4). The term 'software' was introduced in 1959 (Campbell-Kelly, 1995). Companies developed software by direct order for specific customers, and later evolved into developing standard application software for the customer (B2C) and business-to-business (B2B) markets. Table 1 shows an overview of the three business models of software companies (see Valtakosi & Rönkkö, 2009).

Table 1

Three generic business models of software companies

	Supplier of Individual Software	**Supplier of Standard Software B2B**	**Supplier of Standard Software B2C**
Offering	Software developed for the specific needs of a company	Software developed for a mass market on the business side	Software developed for a mass market on the consumer side
Revenue Model	Payment for completed solution or used resources	One-time licence fee plus maintenance fee	One-time licence fee
Distributed Model	Direct contact with customers	Direct contact with customers	Digital or physical retail or pre-installation exceptions on computer hardware

Up to the previous decade, there was some debate about the establishment of software business as a discipline on its own and as a field of research (Mäkelä, 2005; Mäkelä & Mutanen, 2005). Whether software business is a discipline by itself or refers only to the knowledge transfer between software engineering and business management is the crux of this issue (Kontio et al., 2006; Rönkkö, Valtakoski & Peltonen, 2010). Consequently, software business research has not taken off.

The music software industry subscribes to the B2C model, in which software development companies produce music software products for customers. The following subsections review how software production is financed and the success factors of good software development from a business perspective.

2.4.1 Software Development Financing

A software development project would normally be funded for a set period. Software development for commercial releases shows financial constraints similar to those of other industry products. However, the software industry is 'difficult to analyze because it penetrates other industries and evolves rapidly and in unexpected ways' (Käkölä, 2003, p. 1).

Based on empirically tested data taken from an international technology firm, Nam and Harter (2009) found the effects of budget and schedule

pressure on software cycle time and effort to form U-shaped functions, as shown in Figure 9. This means that neither excessive nor limited budget pressure generate a better software development result. The achievement of a potential positive effect on schedule pressure requires collaboration between software developers and stakeholders.

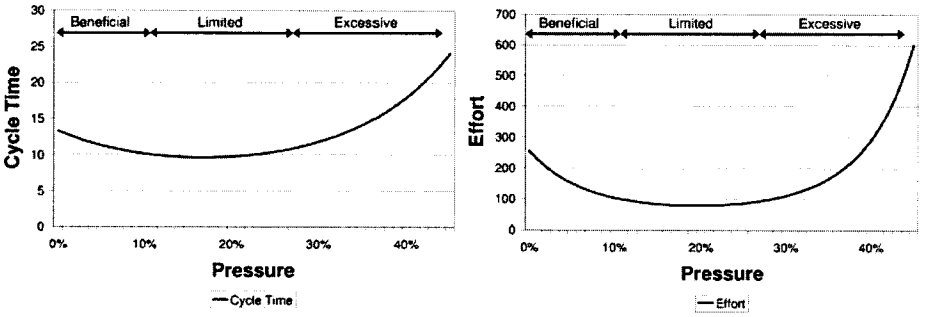

Figure 9. Effect of budget pressure on software development cycle time and effort

2.4.2 Success Factors of Software Business

In his book *The Business of Software,* Cusumano (2004) proposed the following list of basic questions that deal with the fundamentals of products, markets and strategic positioning for software developers to consider during the software development process.

1. Do you want to be mainly a products company or a service company?
2. Do you want to sell to individuals or enterprises, or to mass or niche markets?

3. How horizontal (broad) or vertical (specialised) is your product or service?
4. Can you generate a recurring revenue stream to endure in good times and bad?
5. Will you target mainstream customers, or do you have a plan to avoid 'the chasm'?
6. Do you hope to be a leader, follower or complementor?
7. What kind of character do you want your company to have?

Although these questions seem to be business considerations that are little related to software development, they are essential in the software development process. For example, if a software product is expected to generate a recurring revenue stream (Question 4), its implementation stage should allow room for further refinements, updates or additional features to attract customers to pay the additional charges.

Ahmed and Capretz (2007) summarised a list of the key business factors involved in managing a successful software product line through a literature review (Bayer et al., 1999; Clements & Northrop, 2002; Ebert & Smouts, 2003; Fritsch & Hahn, 2004; Kang, Donohoe, Koh, Lee & Lee, 2002; van der Linden, Bosch, Kamsties, Känsälä & Obbink, 2004; Toft, Coleman & Ohta, 2000). The key business factors included strategic planning, order of entry to the market, brand name strategy, market orientation, relationships management, business vision and innovation. To increase the understanding

of the influence of these factors, the team surveyed eight of the software organisations involved in the business of developing software product lines. The evidence pointed to the key business factors as essential in improving the overall performance of the software development process.

2.4.3 Observations

This section has reviewed how the software industry works, how software production is financed and the key success factors involved in the software business. Echoing Käkölä's (2003) view that the software industry is 'difficult to analyze because it penetrates other industries and evolves rapidly and in unexpected ways' (p. 1), it is not appropriate to view the software business and especially the music software business as conventional production-based businesses, as additional considerations such as educational and artistic needs must be considered.

This study has found no literature related specifically to the music software business. The literature on music, software and business has focused mostly on piracy, copyright issues, digitalisation and compression.

2.5 Ecology of Software Business

The literature review has provided a preliminary model of the ecology of software business (shown in Figure 10).

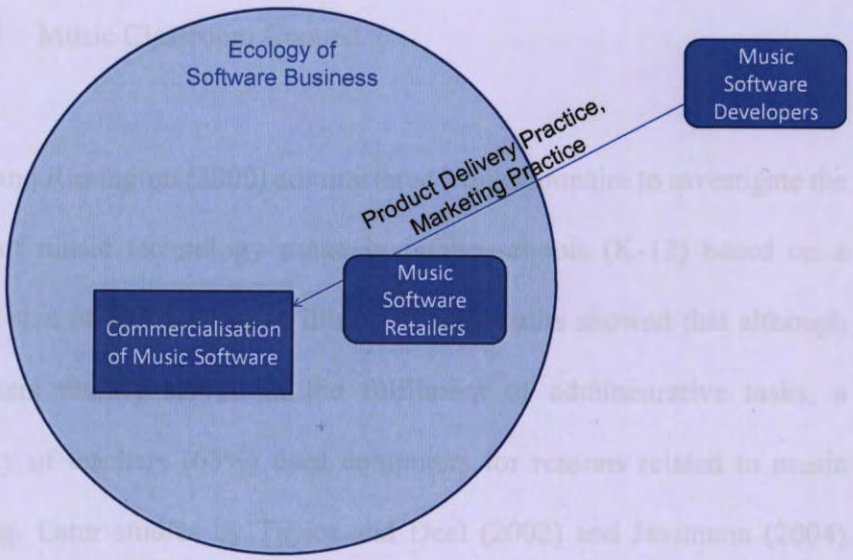

Figure 10. Ecology of software business

Software retailers serve as the bridge between developers and the commercialisation of their software. Putting a software product on the market requires help from the software retailers. A software business must practice good product delivery and marketing strategies to be successful.

A lack of literature on software business has been found, especially those related to interdisciplinarity. Most studies focused on the status of current software business practices and key success factors. More information about the relationships between the ecology of software business and that of music education is required, in addition to a more detailed consideration of the stakeholders in these ecologies. This is provided in Chapter 5.

2.6 Music Software in School Education

2.6.1 Music Classroom Context

Reese and Rimington (2000) administered a questionnaire to investigate the status of music technology usage in public schools (K-12) based on a sample size of 493 schools in Illinois. Their results showed that although computers mainly served in the fulfilment of administrative tasks, a majority of teachers (65%) used computers for reasons related to music teaching. Later studies by Taylor and Deal (2002) and Jassmann (2004) echoed this finding. Only 45% of the schools surveyed in the study by Reese and Rimington (2000) had computers in their music areas or classrooms. The teachers and students mainly used software for word processing, database/spreadsheet, graphics, presentation and Web surfing purposes, with a disappointing 6% using multimedia software.

More recent studies by Dammers and Bauer (2012) continued to reflect on the low adoption of music technology in school music education. Time constraints, insufficient technological competence of music teacher and limitations of schoolbudget were the major inhibitors for the adoption. Music teachers were also found not to be updating themselves with the latest technology, resulting in a gap between their teaching and what was happening outside the school context.

A UK government report recently showed a similar status of music-teaching-related technology usage (Ofsted, 2009, p. 6; 2012, p. 6). Many recent research studies have considered the reasons for this underuse, including a lack of confidence and support from teachers, a lack of pedagogical considerations, teachers' lack of technical skills in relation to music-specific ICT and pedagogical techniques and limited equipment and related resources (Borota & Cencič, 2012; Dorfman, 2008; Gall & Breeze, 2007; Gall, de Vugt & Sammer, 2012; Jautakyte, 2012; Myllykoski, 2012; Roblyer, 2006).

2.6.2 Music Education Applications

This subsection reviews the literature on three music classroom activities, including composition, aural skills and performance skills, in considering the applications of music technology to teaching and learning.

2.6.2.1 Music Composition

Music technology makes two important contributions to teaching and learning: it promotes the widespread use of music composition and music notation software, and makes computer-generated audio playback available through the MIDI protocol. Before the emergence of such technology, music composition was paper based. Composers had to write out everything from clefs to musical notes. They were forced to maintain accuracy in terms

of spacing, juxtaposition and other layout properties unrelated to music making. No cues were given for any composition errors such as out-of-pitch ranges or incomplete bars. The creation of part scores is a labour-intensive process that requires a copyist to copy every part, from the full score to separate sheets. A composer cannot audibly preview his or her work with paper scores alone. How the composition sounds relies totally on the composer's imagination, which is dependent on experience and not favourable to student composers.

In his case studies of computer-assisted composition projects in Hong Kong, Chen (2012) studied how three students with varied backgrounds composed with the help of computer-assisted tools such as *Sonar* and *Finale*. He found that the strategies between and across the creative thinking stages varied among the students. Music technology has a direct effect as a tool during the music composition process, during which its functions vary from recording and refining to improvising and experimenting. Looking at composition strategies, Seddon (2002) and Seddon and O'Neill (2003) studied 48 adolescents who were invited to compose using a specifically modified computer-based composition program. As in Chen's study, the results indicated that the participants adopted different composition strategies. The studies by Seddon (2002) and Seddon and O'Neill (2003) further found that adolescents without prior music training could explore the possibilities made available by the technology more than the musically trained adolescents.

Tobias (2012) conducted another case study to investigate how secondary students engaged in a songwriting and technology course. The study's findings suggested that students engaged as 'hyphenated musicians' by thinking and acting as songwriters, performers, sound engineers, recordists, mixing engineers and producers in recursive and often overlapping ways. Music technology played a role in facilitating a hybrid course that made overlapping and interrelating musical roles and intelligences possible. Collins (2005) gathered similar findings in his 3-year single case study, in which problem proliferation and successive solutions occurred not only in a linear manner but also recursively with the help of music technology such as digital MIDI save-as files and audio files.

Kirkman (2010) examined how individual secondary students developed their compositional skills when working in computer-mediated environments over time within the music classroom context. A key finding of this study was the students' selection of appropriate technologies in different phases of the compositional process. One participant changed his computer-mediated composing environments to apply different technologies to his composition process. This implied that no software is capable of performing all of the tasks required during the compositional process. The study also pointed out that limited classroom resources were the main factor hindering the students' composition processes. Nilsson and Folkestad (2005) examined the creative processes of 9-year-old Swedish

children in terms of computer-mediated composition. The synthesiser and computer software in their study comprised powerful tools that helped the participants express their musical ideas without being formally trained in music. The digital tools used by the children represented a medium where planning, improvising and contingency elements coexisted.

The literature on the use of music software in teaching, learning and practicing composition in a classroom context has unfolded the possibility of multiple perspectives, coexistence and the overlapping of compositional processes and roles. This possibility indicates a hybrid composition process that contrasts with the linear approach of traditional composition teaching and learning. There are always limitations involved in teaching and learning environments and the music software being used. Careful consideration is required to minimise the hindrances created by these limitations and to select the appropriate software for facilitating students' compositional processes.

2.6.2.2 Aural Skills

Hopkins (2002) studied the effectiveness and differences of computer-based expository and discovery methods of instruction for fostering the aural recognition of selected musical concepts. Themes and variations were tested before and after using customised music software on undergraduate music majors. While no significance was found between the two post-tests (one

conducted immediately after the software trial and another after an interval of 6 weeks), the participants who used the discovery method identified more variations.

Cain (2010) conducted an intervention study to systemically teach secondary school students several strategies for improving their listening skills using computer software. The results showed that computers might have helped the students listen to music by enabling them to focus on specific passages, listen several times, gain familiarity with the music and complete tasks (such as notating music) on their own time.

Greher (2004) used a multimedia program to encourage secondary school students to practice critical listening. Participants were presented with software and some music soundtracks and movie clips, and were tasked with matching the appropriate soundtracks with the movie clips according to the moods of the clips. They were also allowed to create their own music and compare it with the original soundtracks. The attitude surveys suggested that the software created an environment that encouraged the participants' active engagement with music.

2.6.2.3 Performance Skills

The i-Maestro project described by Ng (2008) presented interactive multimedia environments for technology-enhanced music education. The

project introduced a system known as 'Gesture Follower' that tracked a performed gesture in real time and compared it with pre-recorded gestures for a variety of pedagogical applications, such as regulating the bowing techniques involved in playing a violin. A modified violin with tracing units attached was connected to the computer software to capture and present the bowing movements of violin players. Hall and O'Donnell (2010) also developed a model for learning violin bowing that could calculate predicted bow speeds and positions after analysing the piece of music.

A recent breakthrough in vocal pedagogy involved the applications of real-time visual feedback technology (VFT) in enhancing vocal skill development. The idea of VFT has been used in vocal pedagogy since Scott (1968) introduced it. However, the technology was not ready to be broadly implemented in general music education until the turn of the century.

Thorpe (2002) pointed out that for VFT to facilitate vocal pedagogy, the technology should be simple to interpret, convey the information in a manner relevant to the required learning task and use a display that relates to some physical aspect of how the associated voice characteristic is produced. Callaghan, Thorpe and Doorn (1999) conducted extensive qualitative and quantitative research on VFT software for vocal training using various approaches that produced a range of findings. In their qualitative study (*ibid.*), they examined how singing teachers incorporated visual feedback software into their singing lessons. Their findings indicated that it was both

feasible and productive to use such technology in the teaching of singing; however, the software was found to be inadequate for specific work on poor-pitch singing. In another study with similar settings, the same research team discovered that teachers were more cautious about students using VFT as a practice tool, emphasising that unsupervised practice with VFT could lead to bad habits if students focused only on improving pitch accuracy (Callaghan, Thorpe, & van Doorn, 2004).

Wilson, Lee, Callaghan and Thorpe (2007, 2008) found that singers whose training combined traditional methods and real-time visual feedback experienced greater improvement in their pitch accuracy than those using traditional means. In terms of the incorporation of such technology into singing pedagogy, the research team claimed that it required the sharing of knowledge across areas such as neuromuscular skill acquisition, information design and processing and musicology. Wilson, Thorpe and Callaghan (2005) assessed the pitch accuracy of singers with varied training experience in singing a sequence of interval patterns with real-time visual feedback provided by Sing and See. Although the beginning singers benefited from the richness of the contextual feedback due to the unfamiliarity of the task, the pitch accuracy of the skilled singers decreased due to the extra cognitive load. The team suggested that the skilled singers experienced a decrease in pitch accuracy because their prior training had the negative effect of an added cognitive load.

Welch, Himonides, Howard and Brereton (2004) conducted an action research project (known as VOXed) to investigate the effect of real-time visual feedback on the learning experiences of singing students. VFT was found useful in improving singing, particularly through the benefits of both real-time feedback and the availability of data capture for subsequent playback and discussion (Howard, Welch, Brereton, Himonides & Howard, 2004). Welch, Howard, Himonides and Brereton (2005) later reported further findings on VOXed, including the differences between individual pedagogical strategies adopted by singing teachers and the increase in the time students spent interacting with VFT during their singing lessons.

Other than *Sing & See* for vocal training, research has also been undertaken to assess the effectiveness of *SmartMusic* – an interactive tool for ensemble skills training and assessment on rhythmic and pitching skills – for instrumental training. *SmartMusic* provides immediate feedback of students' instrumental playing which is essential for helping students to be more engaged in their instrumental learning (Flanigan, 2008; Glenn, 2000; Lee, 2007). Although research examining the effectiveness of *SmartMusic* on students' instrumental skills has not always been positive, advantages have been determined by researchers for its authenticity as a performance assessment tool (Buck, 2008; Lee, 2007; Sidwell-Frame, 2009).

As an assessment tool applied to student trombonists, Long (2011) found that *SmartMusic* only employs a limited range of objective criteria. Its

limitations lies in the exclusion of subjective criteria and other objective criteria. Such limitations include the incapability to measure articulation and note distinction, evaluate the volume of the subject's sound, and recognize visual elements of the student performance that may hinder student progress and pedagogical development – for example, improper breathing habits, incorrect posture and awkward grip of the trombone.

Research has demonstrated the effectiveness of music technology applications in performance skills development. While such applications are rather new to school education, studies have yet to investigate the barriers that hinder schools in applying technology to develop students' performance skills.

2.6.3 Observations

This section reviews the literature on how music technology has been used in the classroom context. The literature has shown that the use of music software is not commonplace in school education, and in-depth research has revealed various factors hindering school music teachers in using technology in their teaching processes. This section also reviews the applications of music software from an activity-based perspective. Three types of music classroom activities, including composition, aural skills and performance skills, have been identified and prove the effectiveness of applying music software to the learning process. Training and assistance are

required to prepare music teachers in applying such technology to their teachings.

2.7 Music Technology in Higher Education

2.7.1 Music Programmes

Meltzer (2001) studied the experiences of music major freshmen and their attitudes towards technology in the US. The findings from the 311 participants surveyed indicated that they were mostly experienced in word processing software and other non-music applications. However, only one third of the participants had experience with music software of various types.

Airy and Parr (2001) investigated student perceptions of the educational usefulness of writing music using MIDI sequencing software packages in a polytechnic institution offering postgraduate certificate and diploma level programmes that specialised in vocational training related to music technology. Their results showed that the participants viewed MIDI technology positively, as it enabled them to compose music without the need for mastery and the different data entry methods catered to individual musical aptitudes. Sound quality was a critical issue, as it fuelled the creative potential for good sound quality. In contrast, bad sound quality was a source of frustration.

Ho (2007, 2009) examined the use of multimedia technology for both undergraduate and graduate music students in Hong Kong. Her earlier study (2007) showed that graduate level music students were confident in their abilities to use multimedia technology, but did not believe that the introduction of multimedia technology into their curricula could improve the quality of their education. According to the results of her later study (2009) on the same issue, which also examined undergraduate level music students, both undergraduate and graduate music students thought that the use of multimedia could raise their interest and enrich their knowledge. The two types of music students differed in their purpose and use of technology. The undergraduate students mainly used technology for school examinations and assignments, and the graduate students used it for academic and teaching purposes in addition to enriching their own knowledge.

2.7.2 Teacher Education Programmes

Price and Pan (2002) studied the framework of technology training in music education degree programmes at 309 colleges in the southeastern US. Their study showed that about 40% of degree-awarding institutions offered one to three technology courses for music education students, and that almost two thirds of these institutions had at least one dedicated lab for music education technology. The most striking finding of the study was the agreement

among music teachers on the importance of self-study in acquiring music technology skills and knowledge.

Bauer, Reese and McAllister (2003) undertook a quantitative study to determine the effectiveness of a 1-week technology workshop series, taken by in-service music teachers as part of a professional development programme to enhance their use of technology for instruction. The results showed that three indicators of effectiveness, including teacher knowledge, teacher comfort and frequency of technology use, were significantly improved after the workshops. However, the follow-up test conducted 9-10 months later showed a significant drop in the three indicators. This highlighted that music teachers required follow-up support to sustain their interest in and use of the technology.

Research has been conducted on the gender differences in pre-service music teachers' relationships with technology. Fung (2003) considered such differences as they pertained to familiarity with technology. The results of the study showed few differences between the familiarity of males and females in terms of the types of technological application. Bauer (2003) examined the self-efficacy and gender differences of technology as they applied to pre-service music teachers. The results of the study showed that experienced computer users were better at self-efficacy, and that males were better at self-efficacy than females. In general, pre-service music teachers showed high computer self-efficacy.

Ohlenbusch (2001) found that pre-service music teachers seemed to use technology more for administrative tasks as opposed to music curricula, echoing Meltzer's (2001) findings on music major freshmen. Gall (2013) surveyed six cohorts of pre-service music teachers (n=93) from the academic years 2006/2007 to 2011/2012 on the factors hindering their use of music technology during their school placements, and found the main inhibitors to be a lack of computers and other equipment and a lack of music staff who were sufficiently competent, confident and/or interested in the ability of technology to provide effective support. Busen-Smith (1999) reported similar findings in a study on PGCE music technology training courses in terms of the factors inhibiting pre-service music teachers in applying music technology in their teaching processes. These factors included a lack of personal ICT skills and confidence, difficulties experienced in integrating music technology and music skills in the classroom, concerns arising from the availability of different computers or software in universities and schools, technical difficulties, anxieties about equipment not working and the problems associated with monitoring student composition work created through sequencing software.

E-portfolios have been widely adopted in music teacher education. They can help pre-service teachers better understand the complexities of teaching, make connections between classroom learning and field experience and develop into flexible and reflective practitioners (Carroll, Potthoff & Huber,

1996; Krause, 1996; McKinney, 1998). Bauer and Dunn (2003) reported on the successful use of e-portfolios in music teacher education:

> *The e-portfolio provides a structure for pre-service teachers to reflect regularly on their strengths and weaknesses as they progress through their undergraduate curriculum ... completion of an electronic portfolio also allows pre-service teachers to be assessed in relation to important teaching competencies in an authentic way. (p.17)*

Berg and Lind (2003) found e-portfolios to encourage reflective and critical thinking on students' goals and progress. They were also found useful for demonstrating the professional skills acquired by students during the course of teacher training. The e-portfolio could become a medium for student learning and self-development, and a means of drawing together the disparate strands of the teacher training programme (Dunbar-Hall, Rowley, Webb & Bell, 2010).

Thornton, Ferris, Johnson, Kidwai and Ching (2011) surveyed six groups of stakeholders (current students, student teachers, alumni, mentor teachers, employers and music education faculty) on their perceptions of an e-portfolio programme at Pennsylvania State University. The alumni seemed to perceive greater value in e-portfolios than the current students. The mentor teachers and employers had minimal interest, awareness and involvement in the e-portfolio process or outcomes. The study results showed that e-portfolios provided students with opportunities to be creative, express their learning and reflect on their strengths and weaknesses through

their teacher training. From a software development perspective, while teacher training is not a problem, the authors concluded that technology frustration was always an overriding theme for students to use e-portfolios, as 'no amount of training or assistance could supplant an intuitive, flexible platform for students to feel they can focus on reflection and personalization' (p. 74). While personal websites have developed tremendously over the past few years, e-portfolio platforms continue to lag behind mainstream platforms in terms of personalised data management.

Pre-service music teachers' attitudes, personalities and motivations for using technology comprise another area of research on technology in music education. Within the framework of the 'Big Five' model of personality, Perkmen and Cevik (2010) studied the relationship between Turkish pre-service music teachers' personalities and their motivation to use CAI. The results showed that out of the five personality dimensions (extroversion, neuroticism, openness, conscientiousness and agreeableness), extroversion, openness and conscientiousness were significant factors positively related to the participants' motivation to use CAI.

2.7.3 Music Technology Curriculum

Boehm (2005, 2007) categorised music technologists (students who are studying or have studied music technology as a single academic discipline at the higher-education level) into five generations. The first generation

comprised the 'experimenters' of the 1950s and 1960s who looked at music and technology and tried to develop their own methods of combining the previously separate disciplines into one. The second generation of the 1970s and 1980s extended the efforts of the first generation to produce, develop and exploit music technology for works of art. The third generation of the 1990s and 2000s studied more than one discipline and had a background in more than one field; they pushed music technology forward as a single discipline and made it an academically viable education and research discipline. The fourth generation comprised the first cohort of students who were able to study music technology within one degree. The fifth generation comprises the students in the late 2000s and 2010s who are studying music technology as a single academic discipline. Although Boehm did not clearly identify substantial differences between the fourth and fifth generations, his identification of the fifth generation was a rather prospective one that made researchers and scholars take notice of the development of the discipline.

Boehm (2005) described the challenges and opportunities involved in integrating music technology as an interdisciplinary study area into an academic-discipline-segregated structure. For music technology to be successfully integrated into higher education, a taxonomy of its issues and the borders of the discipline must be defined. As described by Boehm, interdisciplinarity allows for new combinations of old disciplines and encourages new knowledge and feedback.

Winterson and Russ (2009) conducted a survey to study the experiences of music and music technology undergraduates taking the degree for the first time in the UK. Their results showed that music undergraduates did not identify 'music technology' as one of the most difficult courses. In contrast, the music technology students who were required to take more sub*divided music technology courses indicated 'audio technology' as the most difficult course. Those music technology students had expected more recording practice and less scientific and computer-based theory. This showed that even the students majoring in music technology, which offered an interdisciplinary curriculum in between music and computer science, were not well-prepared or motivated for technical works such as music software development.

Busen-Smith (1999) studied the implication of music technology training in secondary PGCE courses at Kingston University. The PGCE students, most of whom were initially inexperienced in music technology, used it in greater proportion during their teaching practices. They were able to transform their power of self-expression in relation to music technology in the classroom, and their discursive achievements supported their work. However, the case studies conducted a decade later by Welch, Purves, Hargreaves and Marshall (2011) at four UK universities reported that while PGCE students identified the applications of music technology in the classroom as important, the PGCE course did not effectively prepare the students given their increasing reliance on technology in the modern music classroom. This

could be explained by the specifically technology-oriented design of the PGCE courses at Kingston University and the latter study's non-interventional approach.

2.7.4 Transformation of Music Education through Technology

McLuhan (1964) championed the notion that technologies extend our capacities; the telescope enhances eyesight, aircraft extends our ability to travel and photographs store visual memory. The same idea applies to music education. Synthesisers extend the natural range and scope of timbre from musical instruments, amplifiers intensify volume and recording technology stores our audio memories. Technology transforms the way we teach and learn music. For example, it may be used as a tool to raise students' motivation and engagement in learning efficiency (Blakeslee, Brown & Hofmann, 2008); to provide a new model in school music education that allows instrumentally untrained students to assess the computer as a musical instrument (Leung, 2003) or as a practice to cultivate digital musicianship (Partti, in press). This subsection aims to trace the transformation of music education caused or influenced by the integration of technology into the curricula.

In his investigation of the effect of technology in facilitating the teaching and learning of music composition, Beckstead (2001) suggested that technology has not only an efficiency function but also a transformative one.

His view complements a recent study conducted by Wise (2010), in which the potential for transformative change in music education caused by technology integration was perceived. Music teachers are encouraged to look beyond the efficiency of technology and seek ways in which technology can re-evaluate and transform the practice of music education with the aid of machines and electricity.

In his study of technology-based music classes in New Jersey high schools, Dammers (2009) found that the students learned music technology skills and knowledge mostly via a self-study approach. This finding echoed that of Cain (2004), who found that ICT was frequently implemented to facilitate student-centred learning.

Although music education relates more to motor skills and aesthetic learning and less to bookwork, technology advancements have influenced the teaching and learning processes to reach beyond physical and geographical limitations. Researchers have been studying and experimenting on the possibilities of music education in e-learning and distance learning (e.g., Bush, 2001; Finney, 2007; Rees, 2002; Salavuo, 2006; Seddon, 2007). However, Webster (2007) pointed out that little or no research has examined the effectiveness of complete courses or entire degree programmes in music and music education by distance learning.

With the expansion of the involvement of technology in music curricula, the assessment process may require reconstruction as music technology gradually becomes as valid as any aesthetic or instrumental skill (Hodges, 2007). Considerations such as what to assess for a student majoring in music technology beyond any instrument or composition must be investigated. In a qualitative study by Savage and Challis (2002), students considering the use of technology in making music thought that technology could speed up the process. However, they recognised that this did not ultimately equate to better quality.

Through interviews, observations and questionnaires, Wise, Greenwood and Davis (2011) described the perceptions and practices of nine New Zealand secondary school music teachers in terms of music technology and how they changed their work in music classrooms. Pedagogical change and different approaches to music making via means of digital technology were perceived, along with teachers' transformation of their pedagogical approaches from instructivist to constructivist pedagogical philosophies. Ward (2009) also demonstrated and extensively described the constructivist pedagogical approach in his fieldwork project on a secondary school and a primary school that used ICT to help their students compose.

Savage's (2010) study showed different results from those of the previously mentioned study (Wise et al., 2011). The results of his survey of ICT availability and usage in high school music classes across the UK showed

that music education was dominated by conservative uses of ICT that reinforced traditional subject content rather than playing a transformative role. Southcott and Crawford (2011) echoed the view that technology in school music education is mostly recognised as a tool. Savage suggested that rigorous strategies for knowledge sharing in this area are required for the broader potential of ICT in music education to be exploited.

2.7.5 Negative Effects of Technology

Although technology is helpful in traversing human ability, it also raises questions about its applications in education. According to Brown (2007), 'The difficulty in managing this (computers) then falls back on our ability to make sure that we are doing the appropriate thing in the first place' (p. 17). He argued that while technology is enabling the learning and production of music at a much more rapid pace than before, music practitioners may forget the fundamentals of what music should be and how it should be made (composed). For example, although one could easily generate thousands of musical notes in music notation software and generate what may look like a professional compositional work, the software may neither ensure the practicability of the work nor assess its aesthetic pleasantness.

Blind acceptance of technology is another concern related to the use of technology in music education. Upitis (2001) claimed that many government policies on long-term funding for computer networks in schools

were not based on sufficient evidence of the value of technology for student learning, but rather on the belief that computers enhance learning. While time, funding and other resources have been taken from regular music curricula to improve the technological framework of the music classroom, the fundamental questions of whether the reallocation of resources facilitates student learning should not be ignored.

Technology changes the way we teach and learn music, both formally and informally. While music can be acquired or created easily without formal guidance and supervision, the informal learning that results from technology may differ from what one learns in the formal music classroom. Gouzouasis (2005) argued, '[I]f we do not turn the attention of our profession to what is happening in the broad landscape of educational technology/technology education, we will lose yet another opportunity to demonstrate the empirical, paraxial values of music, and all the arts, in general education' (p. 15). This should remind music educators that the integration of technology into music education influences the fundamentals of teaching and learning music.

Beckstead (2001) addressed the negative effects of using musical notation software to learn composition. Almost every piece of musical notation software is based on the notation of Western classical music that originated in Ancient Greece. This not only limits compositional learning to Western classical music, but also confines students to traditional note-oriented

compositions. Beckstead warned music educators to be 'aware of the limitations of such technology and its biases toward discrete, mechanical reproductions and Western classical notation systems' (p. 49). This was echoed at the Tanglewood II Symposium, where it was stated that 'music educators should be concerned about the effect of technology on the loss of cultural diversity' (Palmer & de Quadros, 2012, p. 19).

These warnings suggested that while there are a lot of advantages to be gained from applying technology to music education, care must be taken on how it influences the field and the limitations it imposes. It would be appropriate to conclude this subsection with another quotation from the Tanglewood II Symposium: 'If the current adult generation of teachers of music remains in their comfort zone regarding these issues (technological influences towards music education), the world may soon render them irrelevant to the advantage of technology and innovations, and one might say, the future' (Palmer & de Quadros, 2012, p. 21).

2.7.6 Technology and Informal Music Learning

With the current ease of access to music via the Internet and related technology, the influence of informal music learning on how adolescents learn music has increased. The purpose of this subsection is to review how technology has influenced informal music learning. It focuses on the studies

and observations of researchers who have examined this influential relationship.

Waldron (2009) explored the informal music teaching and learning practices that characterised the old-time online music community using Wanger's (1998) community of practice framework. As a hidden participant-observer, he observed that the participants inside the community were aware of their learning styles and how to learn, adapt and manipulate technology to learn music, interact and share with others. His study not only supported but also further integrated ideas already proposed by music education researchers in relation to online music learning in a community.

Through a case study of a professional composer-sound designer who was 'unable to play a musical instrument in any traditional sense, entirely self-taught as a composer but successful commercially', Savage (2005) raised the need to reconsider the role of technology in music education and expand the aims of music curricula and the possibilities of cross-disciplinary practice. The composer's informal music learning was facilitated by music technology, and the school-based education system created many obstacles to his success.

Finney (2007) described young people's engagement with music inside and outside school curricula in claiming that music education is 'regular, ritualized, spontaneous, irregular and pervaded by ICT' (p. 11). Young

people have sought to sharpen their own musical education and the music educational practices of the future via technological means in the hope that such technology will foster a more democratic future for music education.

Lum (2008) investigated the home musical environment for informal music learning by conducting an ethnographic study to examine the home musical environment of 28 first-grade children in Singapore. The use of media, video/computer games and technological sources such as the Internet were pervasive in the home musical environments of most of the participants. Through this study, Lum cautioned music educators to be aware of the influences of media and technology on students' acquisition of musical knowledge and other needs.

According to Palmer and de Quadros (2012), 'With the advent of programs such as *GarageBand*, young people who grew into a world of advanced computer technology have a different orientation to life; they have their own freshly structured realities' (p. 19). Music educators should not ignore the influence of technology on informal music learning; otherwise, formal curricula would become more distant from the real-world experience of young people in making and enjoying music.

2.7.7 Observations

This section reviews the literature on the use of music technology in higher education and research. The literature has shown that training is required to prepare in-service and pre-service music teachers to use technology in their future teaching and change their attitudes towards the technology. This section also reviews how music technology has transformed music education, its negative effects and its influence on informal music learning. Although the integration of music technology into the music education agenda has introduced enormous transformation, we must take care and make sure that the music technology we are creating and using is facilitating music teaching rather than limiting it.

2.8 Ecology of Music Education

The literature review on music software in school education and music technology in higher education and research has provided a preliminary understanding of the ecology of music education. Figure 11 shows the ecology of music education.

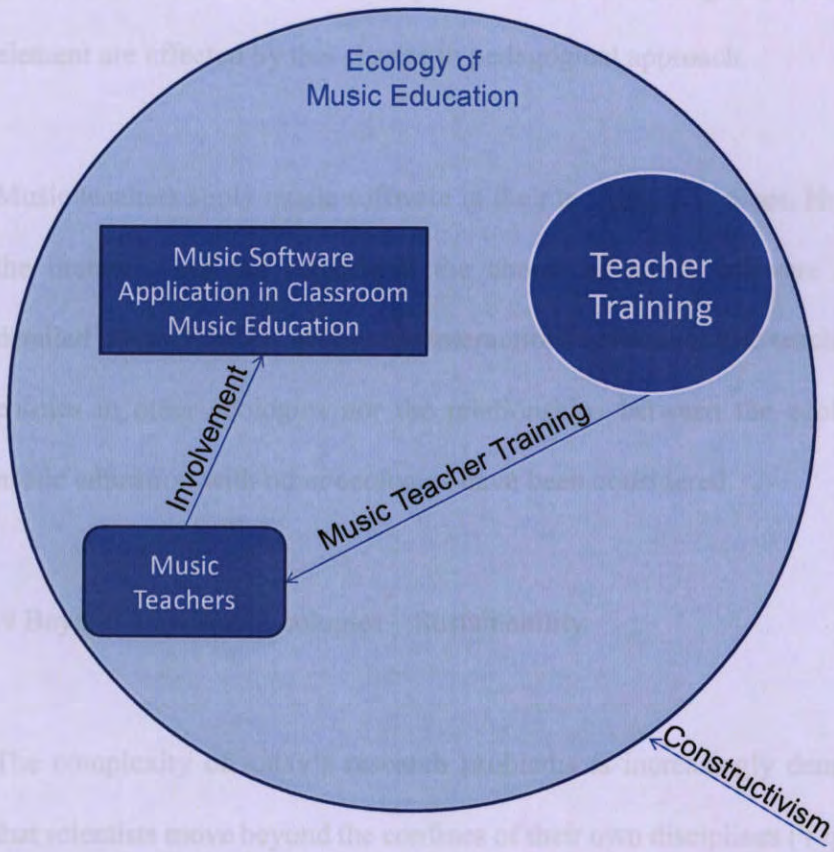

Figure 11. Ecology of music education

Music software usage presents several dynamics for teacher training and music teachers. The first is technological competence, which refers to the training of music teachers to handle and take advantage of technology. The second is pedagogical skill, which refers to the training of music teachers to use software when teaching. The third is positive attitude, which infers that the training programmes must cultivate the positive attitudes of music teachers towards the use of music software. Constructivism as an external dynamic affects the whole ecology as music education shifts from being teacher-centred to student-centred. The applications of music software in

school music education, the discipline of teacher training and every other element are affected by this change in pedagogical approach.

Music teachers apply music software in their teaching processes. However, the literature has not considered the choice of music software and its detailed usage. Further, neither the interactions between music teachers and entities in other ecologies nor the relationships between the ecology of music education with other ecologies have been considered.

2.9 Beyond Individual Ecologies – Sustainability

The complexity of today's research problems is increasingly demanding that scientists move beyond the confines of their own disciplines (Tappeiner, Tappeiner & Walde, 2007). As individual disciplines continue to grow in complexity, an emerging issue is the need for interdisciplinary collaboration which involves knowledge transfer and the consideration for sustainability.

Davis (2010) reviewed the diffusion of IT innovations in education from an ecological perspective by presenting a Venn diagram of the IT innovations in education ecosystems, covering classroom applications to a worldwide scope, as shown in Figure 12. The classroom is at the centre, nested within the school, the region and then the world. Additional ecosystems that overlap multiple ecosystems are also presented, including a college education centre, a multinational IT company and UNESCO. Each of these

overlaps the other ecosystems at different levels. The overlapping ecosystems watch over and facilitate the interactions between other ecosystems from a wider angle. Davis claimed that teachers who adopt IT innovations in the classroom should engage in a bi-directional interaction with the outer ecosystems. Such an interaction would allow teachers to acquire support from the outer ecosystems while also shaping the outer ecosystems and redefining what they should be.

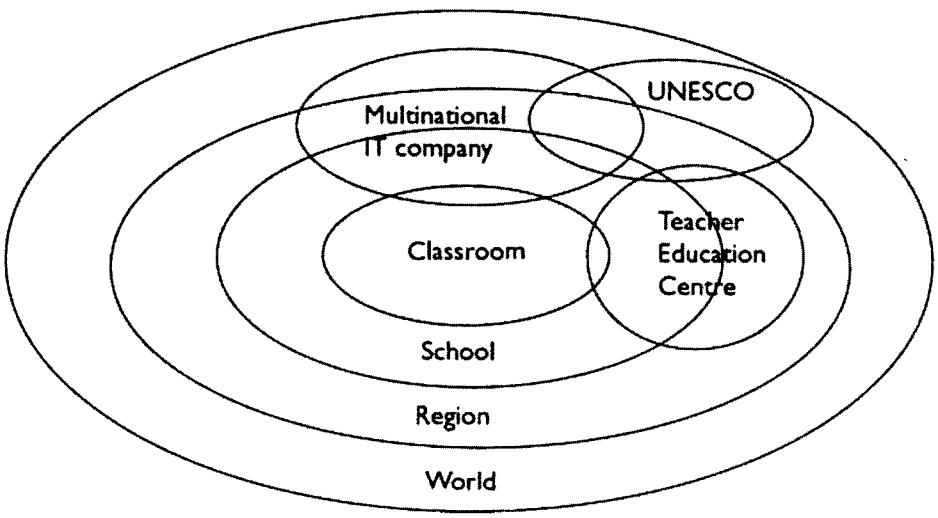

Figure 12. Venn diagram of a teacher's classroom nested within a school ecosystem and regional ecozone in the global biosphere. A multinational IT company, a teacher education centre and UNESCO are used to represent overlapping ecosystems (not to scale)

From an ecological viewpoint, Boyce-Tillman (2004) observed that a narrow focus has been placed on limited aspects of the music experience, ignoring areas such as expressive character, value systems and spirituality

that link the musical experience to the fabric of life lived beyond the confines of the classroom and academe. Boyce-Tillman developed an ecological model that incorporated the notion of 'subjugated ways of knowing' to analyse moves in the educational environment. She concluded that music education in the contemporary world was unable to stand on its own, and that it should interact with other disciplines to achieve a truly musical ecology in response to the pluralist society.

2.10 Summary

This chapter has presented an overview of the research and status of six different areas relevant to the study. The literature review has identified some dynamics of changes in the individual ecologies, particularly in the recent decade. However, collaborations between the different ecologies have been very limited compared with their individual achievements. The advantages, importance and necessity of interdisciplinary teamwork between music software developers and music teachers in developing music software that caters to the pedagogical needs of teachers have been pointed out for more than a decade, and efforts to put this teamwork into practice have thus far been insufficient.

Ecological studies of how music software development works in catering to the pedagogical needs of music teachers have also been lacking, including

studies of the involvement of music software retailers as the communicative bridge between music software developers and music teachers.

This chapter has provided a basis for establishing a sustainable ecosystem consisting of the ecologies of software development, software business and music education. The following chapter will describe the research methodology of this study.

CHAPTER 3

Methodology

According to Hesse-Biber and Leavy (2004), a paradigm may be viewed as a set of basic beliefs that deals with ultimates or first principles. It represents 'the worldview that defines the nature of the world, the individual's place in it, and the range of possible relationships to that world and its parts' (p. 21). A research paradigm not only denotes a way of seeing things, but also constitutes the position that the researcher takes and influences the decisions made during the research process, such as which research method to use (Hesse-Biber & Leavy, 2011, p. 36).

A heavy emphasis was placed on the quantitative approach before the quest of the social sciences became recognised as a discipline. The quantitative approach, also known as the positivist's paradigm, holds that there is a knowable reality that exists independently of the research process. This paradigm fits with the natural science disciplines, which aim to verify a priori hypotheses. When the paradigm is applied to the social sciences, it treats the social world in the same way as the natural world, which is governed by defined rules and patterns. Causal relationships between the variables can be identified, proved and explained, and the social reality can be predicted and potentially controlled (Hesse-Biber & Leavy, 2011, p. 8).

Critics of the quantitative approach in the social sciences have focused on its defects (Hesse-Biber & Leavy, 2004). First, the approach ignores the contextual details of the problem being researched, and detracts from its applicability because the outcomes may only be applied to other problems with very similar environments and situations. Second, just as the quantitative approach does not focus on contextual understanding, human activities and behaviour that can hardly be quantised are not suitable for quantitative investigation. Third, quantitative methods are used to test a priori hypotheses, in which the etic (outsider) theories may not be applied to every emic (insider) situation; this is especially true in the social sciences, where theories should be qualitatively grounded (Glaser & Strauss, 1967; Strauss & Corbin, 1990). Fourth, the quantitative approach assumes that things happen naturally, and so its findings may be recorded objectively and entail a belief that only that which is observable can validly be warranted as knowledge (Bryman, 1988). This is not the case in the social sciences, where findings are created through the interactions between the researcher and the subject. Finally, although theories (or values) and facts are assumed to be independent, such is not the case in hypothesis testing, where facts are always framed within and thus depend on a theoretical framework.

Critics have asserted that the quantitative approach, i.e., the positivist's paradigm that viewable knowledge is always there to be discovered, should not be fully adopted in social science research, especially that related to human activity and interaction. Rather, that kind of research fits better

within the 'school' of social science research that incorporates interpretivism (Halfpenny, 1979; Silverman, 1985; Bryman, 1988).

Schutz (1967) was a leader in the development of the interpretive tradition. He explained that social meaning could not be separated from human behaviour (Nielson, 1990). The interpretive strand focuses on understanding, interpretation and social meaning. Interpretivists believe that reality is relative and multiple, and that knowledge is generated through socially constructed and subjective interpretations via human interaction. Therefore, meaning does not exist independently of the human interpretive process. Rather than testing hypotheses or any other natural science approaches, the interpretive method of inquiry involves exploring, explaining and understanding human reality; thus, the purpose of inquiry is to understand a particular occurrence, not to generalise findings to a population (Farzanfar, 2005).

Qualitative research aims to look at a 'process' or the 'meanings' individuals attribute to their given social situations (Hesse-Biber & Leavy, 2011, p. 45), allowing for 'thick descriptions' of social life (Geertz, 1973). Bryman (1988) further added that qualitative research invariably seeks to go beyond pure description to analyse the environments it examines and commit to understanding events, behaviour, patterns and other dynamics and themes in their respective contexts. Each of these aspects must be considered in terms of the ecologies examined in this study to develop a

model. Qualitative data are non-numerical and can be obtained through a variety of methods ranging from structured interviews to participant observations. For the purposes of this study, qualitative is defined as follows:

> *...a situated activity that locates the observer in the world. It consists of a set of interpretive, material practices that make the world visible. These practices transform the world. They turn the world into a series of representations, including field notes, interviews, conversations, photographs, recordings, and memos to the self. (Denzin & Lincoln, 2011, p. 3)*

There have been numerous attempts to balance the two major paradigmatic positions of quantitative and qualitative research. The rise of the new paradigm, i.e., pragmatism, has appeared to establish the compatibility of quantitative and qualitative research. Pragmatism seeks to combine the advantages and offset the weaknesses between these two research paradigms.

Johnson, Onwuegbuzie and Turner (2007) distinguished three subtypes of research combining the use of quantitative and qualitative approaches – qualitative dominant, pure mixed, and quantitative dominant. This study adopts the qualitative-dominant approach in which one relies on a qualitative, constructivist-poststructuralist-critical view of the research process, while concurrently recognizing that the addition of quantitative data and approaches are likely to benefit most research projects (Johnson et al., 2007, p. 124). In addition to the interviews and questionnaire survey,

this study also includes 'mapping' as a research method (Jackson & Trochim, 2002). Examples of mapping as a research method can be found in recent research (e.g. Evans & Foord, 2008; Lee & Gilmore, 2012).

The aim of this study was to develop a sustainable ecosystem model for software development, software business and music education – areas viewed as three individual ecologies that interrelate and interact. This aim could only be achieved by understanding how each of the ecologies works, and a data collection process that reflected as much as was implemented. The ecologies are socially constructed through human interaction where human activity dominates. As such, the interpretive paradigm was deemed appropriate. A qualitative-dominant approach – which includes a review of related literature, ecological mapping, preliminary modelling based on the literature, studies of three different ecologies and model development of the sustainable ecosystem – was adopted as the method of inquiry to strengthen the study's validity.

3.1 Three Phases of the Study

This study was designed to comprise three phases using a qualitative-dominant approach. Approval was obtained from the Human Research Ethics Committee of the Hong Kong Institute of Education. Table 2 provides an overview of the three phases.

Table 2

Overview of the three phases in this study

	Purpose	Corresponding chapter
Phase I Review of related literature and ecological mapping	● To inform the conceptual framework ● To identify components within and between the ecologies ● To identify relationships and interactions between key components in the ecologies ● Construct preliminary models of three ecologies ● To develop the focus interview questions for Phase II	Chapter 2
Phase II Implementation of four studies and data analysis	● To identify key components in the three ecologies ● To identify the dynamics within and between the ecologies ● To refine the preliminary models from Phase I	Chapters 4-7
Phase III Model development	● To develop a sustainable ecology model ● To show the relationships and interactions within and between the ecologies ● To recommend the qualities of a sustainable ecosystem including addressing threats to the three ecologies	Chapter 8

Phase I involved a review of the literature in order to map the ecologies of software development, software business and music education. It consists of

literature related to the three ecologies as reviewed in Chapter 2, including (1) software engineering education; (2) software development; (3) software business; (4) music software in school education; (5) music technology in higher education; and (6) ecological sustainability. This informed the study's conceptual framework and the research approach, and identified the components of the ecologies and their relationships and interactions. The literature review also provided a basis for establishing the interview questions for Phase II.

Phase II consists of four separate studies; three focusing on the three ecologies and one on domain experts of technology in music education. Data analysis would provide insights into the dynamics within and between the three ecologies and clarify the preliminary models from Phase I.

Phase III involves the development of a sustainable ecosystem for software development, software business and music education, including the key threats to each ecology. The model would provide a meta-view of the ecologies of software development, software business and music education. Recommendations for achieving a sustainable ecosystem consisting of the three ecologies would be provided.

3.2 Reliability

'Reliability' refers to the degree of consistency with which instances are assigned to the same category by different observers or by the same observer on different occasions, among coders and across methods and studies (Bernard & Ryan, 2010; Hammersley, 1992, p. 67). In qualitative research, 'strong interrater reliability' suggests that a theme is not just a figment of one's imagination, and adds to the likelihood that the theme is also valid (Sandelowski, 1995).

Member checking (i.e. interviewees) ensured the reliability of this study. The interview transcripts were returned to the participants to ensure data accuracy and palatability (Stake, 1995). This also gave the participants 'a chance to comment and add materials, change their minds, and offer their interpretations' (Fontana & Frey, 2000, p. 751). Themes with data evidence in each study were checked and endorsed by an experienced researcher.

The results of this study were snowballed from literature review through the ecological mapping, construction of preliminary models of ecologies, three studies on different ecologies and another study with domain experts to provide both insider's and outsider's view. This snowball approach allows the researcher to check the consistency of results from different stages of the snowball process

3.3 Validity

In social science research, 'validity' refers to the extent to which an account accurately represents the social factors to which it refers (Hammersley, 1990, p. 57). In this study, validity refers to whether the themes in each of the ecologies and the dynamics in the overall ecology that emerged were able to fully explain the corresponding human activities and interactions.

All of the key informants in this study were purposively selected because they were experts and practitioners in their respective ecologies. The sampling was valid in the sense that the informants truly reflected the ecological factors. Data analysis was performed based on the interpretation of the findings from the key informant interviews. The interview questions and findings were categorised by themes that captured the action in the ecologies and ensured that the data analysis was self-contained.

The researcher adopted an etic approach to observing behaviour inside the ecologies. An interdisciplinary approach is adopted to examine three ecologies, and findings from studies of the three ecologies supported each other. Data triangulation at the first level was done between the literature review, findings from the study of one of the ecologies and its linkages with two other studies. The second level of data triangulation was done between the findings from the studies of three ecologies, interview data with domain experts and the literature review. These two levels of data triangulation enhanced the validity of the conclusions in this thesis.

3.4 Semi-structured Interviews and Questionnaire Survey

Semi-structured interviews and a questionnaire survey were adopted in Phase II of this thesis. Semi-structured interviews allow new questions to be introduced during the interviews, as interviewees often have information or knowledge that the researcher might not have considered in advance (Hesse-Biber & Leavy, 2011, p. 102). This characteristic favoured this study, as the key informants were expected to be more knowledgeable in their respective ecologies than the researcher. The semi-structured interview approach allowed individual respondents some latitude and freedom to speak about issues of interest or importance while keeping the interviews on track.

A questionnaire survey that complemented the questions asked in the semi-structued interviews was used in the study of the ecology of music education. Music teachers were asked to provide demographic information, self-rated proficiencies concerning their technical competence and confidence in using music software for teaching, music software usage and source of knowledge. This was designed with reference to the literature review and relevant texts such as Converse and Preser (1986) and Fowler (2002).

3.5 Participants

In this thesis, the semi-structured interviews were implemented in four studies with five categories of purposively selected key informants from the three ecologies of software development, software business and music education. These include:

1. Software engineering lecturers;
2. Music software developers;
3. Music software retailers;
4. Music teachers; and
5. Domain experts of music technology in education.

As defined by Bernard and Ryan (2010), 'key informants are 'people who know a lot about their culture and are willing to share all their knowledge with you' (p. 370). Although the ecologies are sizeable, it was easy to identify the stakeholders who were completely knowledgeable about the ecologies they worked in. Moreover, because this part of the study sought to understand human interactions and activities, population size was less important than the participant criteria.

3.6 Data Analysis

Data typically represent reductions of our experience (Bernard et al., 1986). The different individuals interviewed in this study had unique experiences and views on the issues related to their respective domains. However,

patterns were also found among key informants of the same type. After the interview data had been transcribed, data analysis was conducted to search for patterns in the data and for ideas to help explain why those patterns occurred (Bernard & Ryan, 2010, p. 109). In every culture, there can be found a limited number of dynamic affirmations (i.e., themes) that control behaviour or stimulate activity and are expressed as patterns (Opler, 1945, p. 198-199).

Open coding was used in Phases I and II to analyse and interpret findings from interviews with key informants to determine the dynamics in the three ecologies. It consists of literally reading line by line and carefully coding each line, sentence, and paragraph (Charmaz, 2004). The interview data were categorised by themes to capture the action in the ecologies. The human activities and interactions in each of the ecologies may be understood according to the emergent themes. The findings categorised by themes may further be interpreted as the dynamics in the three ecologies.

3.7 Summary

This chapter has presented an overview of the methodology used in this study. The study's research paradigms, reliability and validity are described. The study was undertaken in three phases according to a qualitative-dominant approach that included (1) a review of the related literature, (2) construction of preliminary models; (3) semi-structured

interviews, (4) a questionnaire survey, and (5) the development of a model of a sustainable ecosystem. Data from the semi-structured interviews and questionnaire survey were subjected to data analysis and discussed in chapters 4 to 7.

CHAPTER 4

Study 1: Ecology of Software Development

This chapter describes the planning and administration of the data collection process and reports on the findings of this study specific to the ecology of software development. The chapter begins with an overview of the ecology of software development, followed by the focus questions, which guided the semi-structured interviews. The next section describes the research design of the study, followed by the findings and discussion, and concludes with a summary.

4.1 Overview of the Ecology

Software development is one of the three ecologies examined in this study, concerning 'the tasks necessarily involved in the production of software' (Tsui & Karam, 2011, p. 31). From an ecological perspective, it also involves the training of software developers and the state of practice in software engineering. Sections 2.1 and 2.2 reviewed the related literature, and a preliminary model of the ecology of software development was presented in Section 2.3 (see Figure 8).

The key players in the ecology of software development are software engineering lecturers and music software developers. Software developers

receive their initial training from software engineering education, which has a continuously developing curriculum. The music software development process requires the involvement of software developers and music teachers, in which the latter provide users' feedback and their expectations during the software development process.

The reviewed literature did not find interconnections between the ecology of software development and the other two ecologies. It also failed to clarify the relationships between software engineering education and music software development. With respect to music software development, it was unclear what the key players actually do within the ecology and how they make decisions in the software development process.

4.2 Focus Questions for the Ecology of Software Development

The main purpose of the semi-structured interviews with key informants reported in this chapter was to uncover the missing but needed information concerning the ecology of software development. Based on literature review, the following focus questions have been developed to guide this study.

- What is the content of software engineering courses in higher education?
- How do students apply their software engineering knowledge in real-world practice?

- How do music software developers develop their music software?
- What are the goals and vision of music software from the perspective of developers?
- What are the relationships between music software developers and music teachers?

4.3 Research Design

This study involved two sets of semi-structured interviews with two software engineering lecturers from Hong Kong and three music software developers from Hong Kong, the US and Australia (see Section 4.3.2).

4.3.1 Software Engineering Education in Hong Kong

Software engineering education takes place in the computer science degree programmes offered by university engineering faculties. In Hong Kong, the government funds eight higher institutions overseen by the University Grants Committee (UGC). Six of the eight UGC-funded institutions offer computer science degree programmes: City University of Hong Kong (CityU), Hong Kong Baptist University (HKBU), the Chinese University of Hong Kong (CUHK), the Hong Kong Polytechnic University (PolyU), the Hong Kong University of Science and Technology (HKUST) and the University of Hong Kong (HKU). Postgraduate computer science programmes usually focus more on research than on practice. As the core

training of software developers takes place at the undergraduate level, this study focuses on undergraduate computer science programmes. The programme descriptions and curricula of the six computer science programmes in Hong Kong, with a focus on the software engineering component, are described below. The main sources of information were the official websites of these computer science departments.

CityU – BSc(Hons) in Computer Science

The curriculum of the BSc (Hons) in Computer Science (established in 1987) offered by City University of Hong Kong is composed of a set of highly focused core courses and a wide range of electives, providing broad exposure and choices to students. There are five main subject areas: computer systems, algorithm design and analysis and mathematics, software engineering, seminar series and English communication skills. As one of the focus areas of the curriculum, the software engineering component covers the entire software development life cycle, with emphasis on software design and synthesis, database design and management, and understanding of social, ethical and professional issues.

Apart from the two compulsory software engineering courses – Software Design and Software Engineering Practice – there are four elective courses offered under the software engineering and project management stream: Software Testing and Maintenance, Managing Software Projects, Advanced

Internet Applications Development and Software Quality Management, and Topics in Software Engineering (an elective from another programme).

HKBU – Bachelor of Science (Honours) in Computer Science

Founded in 1988, the Department of Computer Science offers different levels and types of programmes related to commuter science. The curriculum of the BSc (Hons) in Computer Science offered by Hong Kong Baptist University is composed of core courses and major elective courses. The core courses cover key topics in computer science, mathematics and general science and technology. These courses can be further divided into four categories: programming and algorithms, computer and communication systems, software development and professional development.

Two advanced courses – Software Engineering and Software Design, Development and Testing – comprise the software engineering components of the programme curriculum. No elective related to software engineering was identified.

CUHK – Bachelor of Science (Honours) in Computer Science

Established in 1987, the BSc in Computer Science offered by the Chinese University of Hong Kong covers the following areas: artificial intelligence, computer and network security, computer networking, computer-aided design, databases, digital hardware technologies, information systems,

internet, multimedia technology, programming languages, software engineering and theoretical computer science.

The Software Engineering course is the only compulsory software engineering component in the curriculum. Two more courses – Reverse Software Engineering and Advanced Topics in Software Engineering – are electives in the curriculum.

HKU – Bachelor of Engineering (Honours) in Computer Science

Established in 1982, the Computer Science programme offered by the University of Hong Kong is composed of core courses and elective courses. Computer science core courses include Programming, Data Structures, Machine Organisation, Analysis of Algorithms, Operating Systems, Computer Networks, Database Systems, and Software Engineering. Other than one core course, elective courses on Implementation, Testing and Maintenance of Software Systems form the software engineering component of the curriculum.

PolyU – Bachelor of Science (Honours) in Computing

Hong Kong Polytechnic University's first computer studies degree was established in 1983. The current computing programme focuses on the applied nature of computing and IT in the business setting, with strong practical elements and applied systems development. The available course catalogue showed six courses related to software engineering: Foundations

of Software Engineering, Software Engineering and User Interface, Software Engineering, Software Engineering Concepts, Software Testing and Quality Assurance, and Software Requirement Analysis and Specification.

HKUST – Bachelor of Engineering (Honours) in Computer Science

The Computer Science programme offered by the Hong Kong University of Science and Technology is composed of core areas and diverse areas. Core areas include programming, data structures and algorithms, operating systems and software engineering. Diverse areas include but are not limited to databases and data mining, networking, systems software, computer graphics, image processing, medical imaging, artificial intelligence, machine learning, computer vision, computer security and theoretical computer science. There is only one compulsory course, Introduction to Software Engineering, within the area of software engineering.

Table 3 summarises the software engineering courses and nature of the courses offered by the six universities in Hong Kong that currently offer computer science programmes.

Table 3

Software engineering courses offered by Hong Kong universities

University	Number of software engineering courses	
	Compulsory	Elective
City University of Hong Kong	2	5
Hong Kong Baptist University	2	0
Chinese University of Hong Kong	1	2
University of Hong Kong	1	1
Hong Kong Polytechnic University	6	
Hong Kong University of Science and Technology	1	0

Undergraduate computer science programmes include many introductory-level courses to provide fundamental training to students. An introductory-level software engineering course is compulsory for all of the programmes mentioned above, but more in-depth software engineering courses are either elective or absent. Moreover, the semester structure means that software engineering knowledge and practical software development projects must be delivered in less than 4 months.

4.3.2 Participants

Two software engineering lecturers from the University of Hong Kong and three music software developers from Hong Kong, the US, and Australia were purposively selected and invited to participate in the semi-structured interviews.

Software Engineering Lecturers

The semi-structured interviews with two software engineering lecturers aimed to investigate how software engineering knowledge is being delivered in software development training programmes, and how the curriculum is being updated. The University of Hong Kong was selected because it has the longest history of providing tertiary software engineering education locally. It also has a reputation for having a very solid curriculum that has nurtured many software developers in Hong Kong. Software engineering lecturers from the other universities were also approached, but they did not express interest in participating in this study.

The two participants were Mr George Mitcheson and Professor Tse Tsun Him, both from the Department of Computer Science, Faculty of Engineering, University of Hong Kong.

Mitcheson has been teaching software engineering in higher education for many years. Before becoming an instructor, he headed or contributed to the development of a wide range of systems spanning fields such as scientific computation, telecommunications, database management systems, control systems and autonomous robotics and has many years of experience in large-scale software engineering and R&D for real-time systems.

Tse is a professor with research interests related to software engineering, including programme testing, debugging and analysis. He is the director of the Software Engineering Group, and is on the editorial board of several prestigious journals including Software Testing, Verification and Reliability, Journal of Systems and Software, Software: Practice and Experience and Journal of Universal Computer Science. He has been teaching software engineering for many years.

Music Software Developers

The semi-structured interviews with the three music software developers aimed to investigate how music software is developed and its relationships with other stakeholders. One Hong Kong and two international music software developers from the US and Australia were selected for this study. Several music software companies that have developed sophisticated and popular music software were invited to participate in the interviews, but no response was received from those companies.

The three participants were Dr Ann Blombach, developer of *MacGAMUT* (Blombach, 2013); Mr Eric Yung, developer of *AURALBOOK* (Yung, 2013); and Dr William Thorpe, developer of *Sing & See* (Thorpe, Callaghan, Wilson, van Doorn & Crane, 2013). The selected software programmes were based on their applicability to school music education.

MacGAMUT is a drill-based software programme for aural skills training. It provides training for users to recognise intervals, scales and chords, and dictation exercises for rhythm, melody and harmony. Blombach described *MacGAMUT* as 'pedagogically sound' because it provides immediate evaluation and feedback to users with increasing levels of difficulty in each drill component, together with musically meaningful dictation exercises. Blombach, an emeritus faculty member of the School of Music at Ohio State University, developed the software in 1995 and has continued to work on and improve it since her retirement in 2001. More than 25,000 students currently use *MacGAMUT* in hundreds of post-secondary and secondary school music programmes throughout the US and Canada (MacGAMUT Music Software, 2013).

AURALBOOK is an examination-based aural skills training app delivered on mobile platforms. It provides nearly 200 aural test questions for each grade of music examination, including the Associated Board of Royal Schools of Music (ABRSM) in the UK and the Australia Music Examination Board (AMEB) at the Australia and Royal Conservatory of Music (RCM) in Canada. As Yung, the founder of the software company, has a professional background in engineering, the software contains a lot of patented technologies in artificial intelligence and cloud computing. *AURALBOOK* has won a number of awards, such as a gold award in the Hong Kong Information and Communication Technology Awards 2012, a grand award in the Asia Pacific Information and Communication

Technology Awards (APICTA) 2012 and a gold award in the Asia Smartphone App Contest 2013.

The *Sing & See* real-time visual feedback (VTF) software was developed by a research team led by Thorpe at the University of Sydney. VTF is a rather new form of vocal skills training in a computer-mediated environment. When vocal input is received via the microphone, the software detects and displays the pitch (both pitch trace and immediate pitch display), volume/loudness and timbre (in real-time spectrographic display). Thorpe and his research team have conducted extensive research on vocal training based on this software (see Section 2.6.2.3).

4.3.3 Administration of the Interviews

A formal invitation email was sent to each participant, asking them to participate and explaining the purpose of the study and the data collection process. After they had agreed to participate, the interviews were arranged at a suitable time and place. The interviews with the two software engineering lecturers took place in their offices at the Department of Computer Science, Faculty of Engineering, the University of Hong Kong. The interview with music software developer Yung was conducted at his office at the Hong Kong Science Park, while the other two interviews took place using Internet Protocol (VoIP) software via Adobe Connect and Skype. The duration of the interviews ranged from 45 to 60 minutes. Each

interview commenced with a brief introduction to outline the purpose of the study and permission was requested to audio record the interview.

The interviews were transcribed from the audio recordings and double checked by the researcher. An email with the interview transcript attached was sent to each participant after the transcriptions were completed, thanking them for their participation and asking them to confirm the validity of the interview scripts. They were also asked to consent to their identity being revealed in this thesis. All of the participants endorsed the interview transcripts and agreed to their identity being revealed.

4.3.4 Data Analysis

Open coding (Charmaz, 2004) was used to analyse the data in this study. The researcher read all of the interview transcripts and compiled a set of recurring 'themes' in the reports. Themes were given titles such as 'goal and vision of music software' and 'music software development practice', which were then used to categorise relevant statements for the data collection. To provide an example, a statement from a participant is analysed below:

> *...We used an iterative approach – developing the prototype and then getting feedback and then refining it (Sing & See). It's not the traditional design-driven approach. (Thorpe)*

Since Thorpe's comment is about his software development process, it is categorised under the theme of 'music software development practice'. By considering interview findings related to interactions and relationships within and between the three ecologies, the dynamics associated with the ecology of software development were identified and added or amended to the preliminary model of ecology of software development as shown in Figure 8. Detail is described in Section 4.6.

4.3.5 Interview Questions

The questions were grouped into five categories with reference to the focus questions: (1) software engineering course content; (2) students' and graduates' practical experiences with software engineering; (3) music software developers' experiences with software development; (4) the goals and visions of music software; and (5) the relationships between software developers and music teachers.

Software engineering lecturers were asked the following questions.

Software engineering course content

- What software engineering techniques are taught in the software engineering course?
- What particular software engineering processes did you teach in the course?
- What were the sources for updating the course content?

Students' and graduates' practical experiences with software engineering

- How might students put their software engineering knowledge into practice within the course and within the programme?
- How relevant are the course materials to real-world practice in the software industry?

Students' and graduates' practical experiences with software development

- How relevant is software engineering education to music software development?

Music software developers were asked the following questions.

Music software developers' experiences with software development

- What software engineering training have you received?
- What motivated you to develop the software?
- What software engineering techniques did you use in developing your software?
- Did you conduct any requirement analysis?
- Did you conduct any software testing?
- What were the biggest challenges in developing your software?

Goals and visions of music software

- What are the goals and vision of your software?
- Who are the targeted users of your software?
- What are the competitive advantages of your software?
- Is your software teacher-centred or student-centred?

Relationships between software developers and music teachers

- Through what channels do you sell your software product?
- How do you advertise your software?
- How do you provide user support to your customers?
- How do you gather feedback from software users?
- What problems have users encountered when using the software?

Others

- What constitutes good music software development?

4.4 Findings

The data from the semi-structured interviews were categorised into 11 themes:

1. Course content of software engineering education
2. Students' practical experiences within the programme
3. Graduates' practical experiences in the industry
4. Technical background of music software developers
5. Goals and vision of music software
6. Music software development process
7. Competitive advantages of music software
8. Sales and marketing of music software
9. Communication with end users
10. Music software development practice
11. Pedagogical considerations

Theme 1. Course content of software engineering education

Although they use different strategies in their teaching, both of the participants said that they introduce the full spectrum of software engineering activities, from project management, software design, requirement analysis to software testing and maintenance. Given the short time of the introductory course, the teaching on those general software engineering activities could only be descriptive.

> *It's a broad course, which teaches the full spectrum of software engineering activities. So there are some aspects of project management, some software processes, principle activities, that form the software engineering process, those tasks which form the activities, and the introduction of a number of special techniques in the areas of design, analysis, requirement capture, (software) testing. (Mitcheson)*

The software development process is the core part of software engineering. Mitcheson responded that he introduces a range of software development processes from the classic waterfall model, the unified process that is the current software engineering approach, and other short iterative processes such as the extreme models. The idea is to teach students 'vocabularies', to help them understand the terminology in the field as an introduction to software engineering practice. Taking a more focused approach, Tse responded that he mainly introduces the unified process together with the unified model language (UML). Other models mentioned by Tse, such as the waterfall model or extreme programme, are either too outdated or too

extreme to be commonly used today. Although not all of the activities of the unified process are taught, Tse's software engineering course only focuses on software design, analysis and testing.

> *We introduce a range of software processes from the classic waterfall model, the unified process which is the current approach of the software engineering process, and other processes such as the very short iteration processes, such as the extreme models. (Mitcheson)*

The methods of assessment for both of the participants' software engineering courses vary each year, but always maintain a balance between examination and project. A within-course software development project is discussed under Theme 3. Tse likes to add some real-world ingredients to the project, such as a sudden requirement change from the 'client' one week before the project deadline, as follows:

> *Actually what we do is, one week before the deadline, we tell them 'we as users have changed our minds, please follow the changes'. It's real life. (Tse)*

While the core of the software engineering principles remains stable, new techniques and trends emerge every year and so the software engineering courses need to be updated frequently. Tse mentioned a list of sources for the course updating, including published books, papers and UML standards. Mitcheson shared his similar experience, putting it this way:

> *For example, a decade ago design patterns became popular with the publication of the famous design pattern book, so we started to*

> *introduce the idea of design patterns with some explanation. More recently, in fact, a new approach was published which built on the idea of the design pattern but took it on to a more environmental level, showing the elements used to construct the design patterns themselves. So I have incorporated this idea. (Mitcheson)*

Tse also gains updates from industrial software projects and contacts with former students. Both of the participants mentioned that course updates are incremental, in which slightly changes appear every year based on the existing theories and materials, without too many big leaps.

Theme 2. Students' practical experiences within the programme

All software engineering courses include a group project that simulates a software development team project in real-world practice. Due to the time constraint, only a small project or part of a big project can be executed by student groups. For example, while Mitcheson's approach is to implement a small software development project using Ruby on Rails – an open source web application framework, Tse asks student groups to work on software design and analysis without implementing the whole system. His rationale is as follows:

> *We do not ask them to implement the system, so they just use IBM rational to draw the analysis and design, diagrams and that's it. Because I feel that they already have a lot of programming courses, I don't need to teach them those programming methodologies. It's also because of the time constraint. (Tse)*

Apart from the software development team project within the software engineering course, all computer science related degrees have an honours

project in the final year. Most students choose to work on a software development project where they can further execute their software engineering knowledge. However, final year students without real-world experience do not appreciate the practice of software engineering, and only implement their designs in practice if required to do so by their supervisors. Hence, which software engineering standard the students apply is dependent on their supervisors. This was reflected by both participants.

> *If they are working under me then yes (they'd make use of software engineering), but if they are working under some other supervisors who themselves are not aware of UML or those kind of standards, then it may not be the case. So it depends on the supervisor. (Tse)*

> *I expect students to test their software and apply standards to their software testing. (Mitcheson)*

However, Tse mentioned that one of the disadvantages of students' honours projects is that the product is only a one-off product. Unlike real-world projects, there is no re-use of codes and documentations, or upgrading the software based on the same software architecture. Thus, there may be not much difference between a final year software product with good software engineering and one without it.

Theme 3. Graduates' practical experience in the industry

Software engineer is a profession. Unless self-employed, most software engineers are graduates from the same discipline in higher education. Both participants highlighted the relevance of their software engineering courses

to real-world practice. Mitcheson said that fresh graduates can enter a new job and start being productive very quickly because they are well equipped to start work immediately:

> *We find our graduates can enter and start being productive in a new job after graduation very quickly because they have knowledge about what's happening there, what they are asked to test, what that means, what documents they have to produce, what resources they need to do the testing. (Mitcheson)*

Tse responded in a similar way by pointing out that governmental and semi-governmental bodies, large software companies and organisations all use software engineering standards such as IBM Rational, which is a compulsory part of the software engineering course.

However, Mitcheson also highlighted that the software industry in Hong Kong generally neglects the importance of software engineering. Software developers are not convinced of its importance, which is reflected in their budget allocations. This could be explained by the working style in Hong Kong – rushing to meet deadlines, looking for new projects and focusing on the quantity rather than the quality of software. Mitcheson put it this way:

> *You go in quickly, you think you understand the problem, you go away and build something that you think can solve the problem, and then you fix it when it's wrong. That's still the attitude in Hong Kong. (Mitcheson)*

Theme 4. Technical background of music software developers

Thorpe and Yung both received their engineering training from the undergraduate degree programmes at their local universities. Although the training they received was not exactly computer science or software engineering (they are electronic and electrical engineering graduates), Yung reflected that the foundation of engineering training provided relevant knowledge for software development. Blombach is a self-trained software developer. She only studied one computer programming course at college, and in her bachelor's degree she majored in mathematics, where the computational calculations were done on punch cards and it took a week to get the results.

Theme 5. Goals and vision of music software

According to Yung, the goal of *AURALBOOK* is to help teachers to teach aural skills in a better way. He observed that teachers today do not have sufficient time or training to teach aural skills in music lessons and the *AURALBOOK* software can help.

> *...to help the teachers to teach aural skills in a better way. Because in the current teaching environment, teachers do not have time to teach aural skills in musical instrument lessons, and most of the time is spent on teaching instruments. (Yung)*

MacGAMUT is also aimed at aural skills training and, as Blombach mentioned, it is intended to 'provide students with a tool to develop aural skills on their own'.

As vocal training software, *Sing & See* is intended to 'improve teachers' and singers' vocal training experience'. *Sing & See* originated from a research project investigating the use of real-time visual feedback technology in vocal training, so it has a vision to research the effectiveness and efficiency of incorporating visual feedback training in vocal training.

All three participants agreed that their software was designed for self-learning, with students as the main targeted users. However, they also pointed out that teachers and parents are stakeholders in the software because, as Blombach mentioned, they can 'change a lot how the software operates'. Yung added that because of insufficient time in school music lessons, teachers can introduce the software to students, who can then continue to learn aural skills on their own at home.

Theme 6. Music software development process

With sufficient software development support, *Sing & See* was developed using an iterative approach, with a prototype developed at the beginning for refinement. Rather than the traditional design-driven software development approach, *Sing & See* was developed iteratively from its prototype.

> *We used an iterative approach – developing the prototype and then getting feedback and then refining it. (Thorpe)*

AURALBOOK was developed in a rather straightforward way. Because the training tasks in this software were based on examinations from different

authorities, the development team used the syllabi as a requirement to develop the software.

> *...our software must fulfil the examination syllabus of the authorities, that's the guideline already. It sets out all the steps, even what they say, how they ask, it is all set. (Yung)*

Rather than using a development model, *AURALBOOK* was developed by direct coding of the programme. The development team consisted of several programmers with different expertise, with none of the team members leading the others in the software development process.

MacGAMUT was developed by Blombach herself as sole proprietor for over ten years. As Blombach is a self-trained software developer and she was working on her own, *MacGAMUT* was not developed using a software development model. However, she did keep very good software development conduct – only three bug reports have been presented since the software was developed. Instead of modelling, Blombach clearly lays out the tasks to be completed before each coding process.

> *I try to lay out very clearly what I want to do before I start programming because otherwise, it wastes a whole lot of my time. But that's the closest I get. (Blombach)*

Theme 7. Competitive advantage of music software

All three participants were asked about the competitive advantages of their software compared with music software of the same type. Yung responded

that the aural skills training apps currently available in the market focus on very minimal types of ear test-sets without providing training on singing, clapping and other simple tasks, whereas *AURALBOOK* provides more comprehensive aural skills training because of its focus on international examinations.

> *First of all, how to define aural skills apps? I know there are lots of ear test-sets. In an ear test-set you listen to something, and then you decide if it is a C major chord, or a crotchet or a minim. That's it. Can this be classified as an aural skills app? From my point of view, it is not. It can be called a grade one music theory app because it teaches you basic skills of music. It is still teaching, but not about aural skills, which are about singing, clapping and listening, with lots of music knowledge, music history etc. So up to now, I haven't seen any app in the Appstore that is directly competing with us. (Yung)*

Blombach responded that the competitive advantage of *MacGAMUT* is the procedure used to train users' aural skills. The *MacGAMUT* system requires users to complete one level before going on to the next, and a level may have more than 1,000 exercises.

Thorpe pointed out that the competitive advantage of *Sing & See* is the pitch recognition algorithm, which provides accurate feedback on the singing voice. *Sing & See* was developed in collaboration with singing teachers, which makes it easy to use.

Theme 8. Sales and marketing of music software

All three participants said that they use online marketing to advertise their software products. Due to the research focus of *Sing & See*, Thorpe also advertises *Sing & See* in singing education journals. Other than Facebook, email and other online marketing campaigns, Yung also makes use of traditional marketing strategies such as sponsorship of concerts to advertise *AURALBOOK*. In contrast, Blombach prefers to use traditional mail to advertise *MacGAMUT* as she believes that most people ignore email advertising nowadays.

> *Yes, we do online marketing, and we also do some advertising in singing education journals. (Thorpe)*

> *...online - which means Facebook, email or any online marketing campaign, or offline - advertisement, sponsorship of some concerts. (Yung)*

> *We like snail-mail, primarily because most people ignore email these days...And we also mail to high school teachers who teach music. (Blombach)*

Sing & See and *MacGAMUT* also offer demo versions as another marketing strategy to give potential users a taste before they purchase. Blombach also arranges many advanced placement workshops in high schools and colleges, which help to advertise *MacGAMUT*.

AURALBOOK is a free app that can be downloaded and used freely and the company earns money by selling additional e-books of *AURALBOOK* exercises. As it was developed as a mobile app, AURRALBOOK can be

downloaded through mobile app markets such as the App store on the iPhone mobile platform and Google Play on the Android mobile platform. *MacGAMUT* is sold in two ways – a physical package available through bookstores, or online from its official website. Thorpe has tried to collaborate with retailers in selling *Sing & See*, but this was unsuccessful because the software is a niche product. *Sing & See* is currently sold online. Both physical and virtual packages are available for *Sing & See*.

Theme 9. Communication with end users

There is a two-way communication process between music software developers and their end users: the developers provide support to their users, while the users provide feedback after experiencing the software. All of the participants responded that email is the main communication channel between them and their software users. Thorpe uses more advanced tools such as Fogbugz and Copilot to communicate with users, as they allow him to work through the users' computers. Blombach uses more face-to-face approaches, in which she works closely with students and teachers to receive suggestions for improving *MacGAMUT*.

> *From time to time, I also work closely with individual students, not just instructors, because a student will suggest something good, and then if I encourage that student, he keeps suggesting things. (Blombach)*

> *We do user surveys, so we have feedback from users. Also, the comments that people volunteer, and interviewing and talking with users one to one. So it's feedback from users generally. (Thorpe)*

Both Yung and Thorpe said that most enquiries from users are about standard technical problems with using the software, such as misconnection with the server, or audio unit malfunctioning. Yung explained that the majority of users' technical problems arise because the users of *AURALBOOK* are music teachers and students who may not be good at technical configuration. With fewer requirements on audio configuration, Blombach receives very few user enquiries and most of them relate to very low-tech problems such as difficulties in downloading the software from the official website, which have nothing to do with the software.

> *...it's usually someone who doesn't know how to download a file from the Web or doesn't know how to put a CD into a CD drive - almost never something to do with the software itself. (Blombach)*

Theme 10. Music software development practice

All of the participants were asked about their views on music software development practice as an interdisciplinary area between music and software development.

Although the two software engineering lecturers are not music professionals, they were asked about music software development from the perspective of software engineering for the purpose of this thesis. Both participants reflected that many problems in general software development also occur in

music software development. For example, Mitcheson pointed out the lack of communication between music teachers and software developers:

> *...most of the problems they face aren't technical. They come from misunderstanding requirements or poor requirement engineering. (Mitcheson)*

Tse replied that 'students like to make up a lot of additional requirements, which are not required by the users' – i.e., 'gold plating' in software development terminology. He added that while music teachers do not know about the technical aspects of software development, software developers have a responsibility to elicit requirements from them in an appropriate way, i.e., by asking them about music-related requirements rather than functional requirements.

As a music software developer, Thorpe pointed out that interaction between software developers and end users (i.e., music teachers and students) is very important for successful music software development, which focuses on the actual software usage by students and teachers. Blombach was also concerned about the importance of addressing students' and teachers' needs in her experience as a software developer.

> *One of the things I notice is the interactions with singing teachers and music teachers. Their priority of focus is on students and on the music. So any software tool they use has to fit in with the actual music, and it's got to be very simple to use – they want to focus on the students and the music rather than on the counter. (Thorpe)*

> *...I taught theory and ear-training for 25 years at Ohio State, and tested it (MacGAMUT) with my students for many of those years. I know it works. (Blombach)*

Yung, in contrast, pointed out that team cooperation is most important. During the development of *AURALBOOK*, there were many technical difficulties, such as restrictions on iOS software development, the shift from the iOS platform to the Android platform and configurations for different hardware. However, collaboration allowed the knowledge and attitude of the team to overcome these technical challenges.

> *I will tell you that the team is the most important. Because the hardware is on the market, any system or software can be changed at any time, but the attitude of the team members and their knowledge is the most important. (Yung)*

Thorpe agreed that good teamwork is the most important factor, but added two other factors: the design of the interface, which affects the ability of users to make use of the software, and the software architecture that allows the software to be easily improved and upgraded. As the sole developer of *MacGAMUT*, Blombach pointed out the importance of time management and the gatekeeping of the software, which provides a bug-free learning environment for software users.

> *And if someone reports a problem (with MacGAMUT), I am almost always much more concerned about it than they are...I've heard of other software developers who don't even take bug reports seriously when they're beta testing, let alone when the real thing is out there. That will never happen with MacGAMUT. I know there's no such*

thing as perfect software, but I do my very, very best to come as close as possible. (Blombach)

Theme 11. Pedagogical considerations

As the three music software packages were designed for educational purposes, the participants seriously consider the pedagogical issues in the development process. For example, Blombach works closely with students and teachers using *MacGAMUT*, so that the requirement analysis process works almost in parallel with the implementation of the software. Thorpe's pedagogical consideration is in students' individual vocal training, whereby students can learn to sing by themselves outside the classroom. Addressing a similar pedagogical issue, Yung focuses on providing a tool for students' self-learning in aural skills development.

As the music software developed by all three the participants is student-centred, it is not conducive for use in the classroom context. Thorpe mentioned that it is meaningless for a group of students to use the software for self-learning in a class together, when they can do the same thing at home. However, Blombach pointed out that guidance is needed from teachers as it could alter the way students use the software to learn. Yung said that *AURALBOOK* is not as complete as he would like and there is room for improvement, such as developing the software for other platforms. He will consider collaborating with schools once the software is completely developed.

4.5 Discussion

4.5.1 Curriculum Development

The two software engineering lecturers said that their course content was updated frequently in response to the rapid changes in the industry. This echoes with Huang and Distante's (2006) emphasis on including real-world software development elements in student assignments, and with van Vliet's (2006) suggestion to reconcile software engineering with real-world practice. However, the semester-based course structure and tight programme content restrict what can be done in software engineering education, leaving very limited scope for what the software engineering lecturers can do.

A better integrated curriculum is needed to provide opportunities for more innovative pedagogical approaches and interdisciplinary software development projects over a longer time-span. Different courses could be strategically structured so that the course content and coursework can be interrelated, allowing students to understand how things work together. This could include the co-implementation of software development projects across several courses, such as programming and software engineering, to put the content knowledge into practice within the same project. The literature review in Section 2.1.1 identified many innovative pedagogical approaches in software engineering education, and demonstrated the

effectiveness and feasibility of those approaches. Such approaches were rarely found in the interview data. To prepare students to 'become agents of change in the industry' and 'stay current in the face of change' (Garlan et al., 1997; Shaw, 2000), there is a need to include more innovative pedagogical approaches to maximise the output of software engineering education within a limited time.

4.5.2 Software Development Training

The software engineering lecturers and the literature recognise the importance of teaching the software development process in computer science programmes, as software professionals place high priority on learning the software development process (Lethbridge, 2000).

Both the interview data from the software engineering lecturers and the analysis of the software engineering courses offered by local universities (see section 4.3.1) revealed that students taking such courses receive only an introductory level of software engineering knowledge. This could be explained by curriculum time constraints, as reflected by the participants, who said that they were unable to cover all of the components of the software development process. It is disappointing that despite the high recognition of its importance, software engineering education does not receive enough attention in computer science programmes. This means that some of the important components, such as quality assurance activities, are

likely to be neglected in software engineering education (Bareiss & Katz, 2011). The implication is that software engineers who graduate without sufficient training in software quality assurance will fail to implement the quality assurance process in their software development, which leaves the quality of software products at risk.

The software engineering lecturers in this study also revealed that requirement analysis is another software development activity that is not being seriously addressed. Software developers add a lot of unnecessary features which make the software more difficult for non-experts to use the software. Viewing the development of such digital products as a design discipline is still uncommon in the industry, where the construction-based view is still dominant (Löwgren & Stolterman, 2004) and the requirement analysis process for understanding users' needs is not seriously addressed. With respect to music software development, this prevents music software developers from developing software with better interactions with users (Cooper, Cronin & Reimann, 2007), or results in the development of music software that is too complex and technical for non-expert users (Lenberg, 2010).

More attention should be given to the software engineering components in computer science programmes, including the allocation of more time and resources to improve software engineering education.

4.5.3 Adherence

Adherence refers to the need to follow the formal procedures of the software development process. Because of the lack of formal software engineering training and the small size of their software teams, the three software developers in this study did not strictly follow the formal software development process. Rather, they devised their own ways of developing their software, which compensated for the lack of formal software engineering training.

Nevertheless, there are drawbacks in veering too far away from the formal procedures of software development (Flores et al., 2001). For instance, the developers may put insufficient effort into parts of the software development process that they are not good at, or they may intentionally neglect the process. Adherence to the music software development process is needed to ensure the software product is authentic and high quality.

4.5.4 Insight and foresight

The music software developers in this study possessed insight into the particularities of aural and singing skills in developing their music software. They understood the needs of the end users, and therefore addressed the issues of student-centred learning and the problem of time constraints in music lessons.

The music software they developed was designed to facilitate students' self-directed learning, so that students can learn in their own time with minimal guidance from music teachers. Student-centred music software can enhance students' learning and change the way music is taught and learnt in the music classroom. This is consistent with the literature suggesting that music technology can not only enhance efficiency but can also be transformative (Beckstead, 2001; Wise, 2010).

The music software developers also possessed foresight in addressing the effects of informal music learning, which echoes with the finding in the literature that music technology can facilitate informal music learning (Finney, 2007; Lum, 2008; Palmer & de Quadros, 2012; Savage, 2005). Most of today's applications are based on concepts originally developed in professional music studios, which do not consider the needs of users outside the profession. While the software market is becoming increasingly competitive, with more products and lower prices (Lenberg, 2010), the increasing need for informal music learning, self-directed learning or any other educational purposes should be addressed in the software development process to attract inexperienced users. Addressing the issue of informal learning could help software companies and developers to develop new user groups, which could help them to survive in the increasingly competitive market with more products and lower prices (*ibid.*).

4.5.5 Value-added involvement

The interviews with the music software developers showed the importance of considering music teachers' needs. These developers worked closely with music teachers and the curriculum in developing their music software. It is gratifying that music teachers' needs were addressed in the music software development process.

As music software developers may not possess the domain knowledge and pedagogical considerations that music teachers have, the involvement of music teachers in the software development process is necessary to fill the gaps between developers' lack of domain knowledge and the contextual usage of music software in education. The involvement of music teachers not only provides users' feedback and experience, but also domain knowledge and pedagogical considerations in the music software development process.

Requirement analysis is the key process for ensuring music teachers' value-added input. Software development training needs to train software developers to perform better requirement analyses to capture music teachers' value-added involvement. This again relies on better software development training, as mentioned in Section 4.5.1.

4.6 Summary

The findings of this study were applied to the preliminary ecology model in Section 2.3 (see Figure 8). The refined model captures the dynamics of the ecology of music software development revealed in the interview data. The refined ecology of music software development is presented in Figure 13.

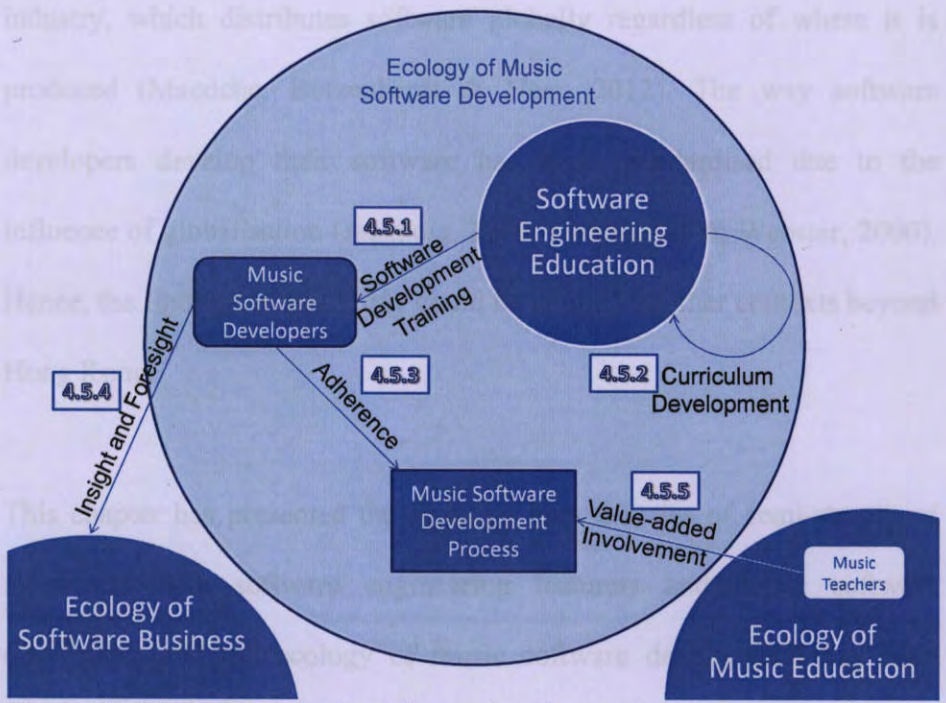

Figure 13. Ecology of music software development

An ecology of the music software development process was developed, as shown in Figure 13. It describes the relationships between the key players and the dynamics in the ecology of music software development, and their relationships with the other ecologies in this study. The numbers in Figure 13 correspond to the relevant sub-sections in the Section 4.5.

Globalisation affects the ecology of music software development in terms of both software engineering education and the software industry. Because of globalisation, software engineering education has become somewhat standardised across the world (Grega, Kornecki, Sveda & Thiriet, 2007; Harris, Tharp & Zalewski, 2002; Lethbridge, Díaz-Herrera, LeBlanc & Thompson, 2007). Similarly, globalisation has also influenced the software industry, which distributes software globally regardless of where it is produced (Maedche, Botzenhardt & Neer, 2012). The way software developers develop their software has been standardised due to the influence of globalisation (Jaakkola, 2009; Lucena, 2006; Webster, 2000). Hence, the findings of this study could be applied to other contexts beyond Hong Kong.

This chapter has presented the findings from a series of semi-structured interviews with software engineering lecturers and music software developers, and the ecology of music software development has been clarified. The ecology describes the relationships between the key players and the dynamics in the ecology of music software development, and their relationships with the other two ecologies in this study. It is recommended that music software developers reach outside their own discipline and become involved in the interdisciplinary cross-ecology music software development process.

The findings of this study show that software development is not a stand-alone ecology. The next chapter, which presents an ecological study of the software business, shows how these ecologies link together.

CHAPTER 5

Study 2: Ecology of Software Business

This chapter describes the planning and administration of the data collection process and reports on the findings of this study specific to the ecology of the software business. The chapter begins with an overview of the ecology of the software business, followed by the focus questions used to guide the semi-structured interviews. The next section describes the research design of the study, followed by the findings and discussion, and concludes with a summary.

5.1 Overview of the Ecology

The software business is one of the three ecologies examined in this study and refers to the commercial activity of the software industry aimed at producing, buying and selling software. Section 2.4 reviewed the related literature, and a preliminary model of the ecology of software business was presented in Section 2.5 (see Figure 10).

The key players in the ecology of the software business are music software retailers. Through music software retailers, music software developers are able to put their software product into the market – also known as the

commercialisation of music software. Good product delivery practices and marketing strategies are needed for a software business to be successful.

There is a dearth of research investigating the relationships between the software business and other areas, and also the relationship between music software retailers and music teachers. From a meta point of view, the role of music software retailers with respect to the ecosystem consisting of the three ecologies is unclear.

5.2 Focus Questions for the Ecology of Software Business

The main purpose of this study was to uncover the missing but needed information concerning the ecology of the software business. Based on the literature review, the following focus questions have been developed to guide this study:

- What are the characteristics of the music software business compared to other forms of business?
- How do music software retailers interact with music software developers and music teachers?
- What information and knowledge are shared and transferred between music software retailers, music software developers and music teachers?

- What are the trends of the music software business from the perspective of music software retailers?

5.3 Research Design

This study involved semi-structured interviews with two music software retailers from Hong Kong and the UK.

5.3.1 Participants

Two music software retailers were purposively selected and invited to participate in the semi-structured interviews. The two participants were representatives from large music software retailers in Hong Kong and the UK: Mr Kenny Lui, Manager of the Computer Audio department of Tsang Fook Piano Company Limited, Hong Kong; and Mr Javan Green, Shop Manager from Millennium Music Software Limited, Nottingham, UK.

Founded in 1992, Millennium Music Software has been selling music technology and studio equipment, specialising in music software for home use to professional studio setups, for more than 20 years. It is currently the retail and mail order music technology specialist in the centre of Nottingham, and is the UK's leading music technology supplier. Its product brands include all major titles, such as Yamaha, Korg, Roland, Alesis and AVID.

Tsang Fook Piano was found in 1916, and is one of the oldest music centres in Asia. In addition to providing music services and products, Tsang Fook Piano specialises in selling music software products in Hong Kong. It offers computer music software products, including all international brands such as AVID, Ableton, Cakewalk and Propellerhead. It also holds music technology workshops for music teachers, educators and students.

5.3.2 Administration of the Interviews

A formal invitation email was sent to each participant, asking them to participate and explaining the purpose of the study and the data collection process. Both interviews were undertaken in the participants' offices at the software retailers' headquarters. The interview with Lui was conducted at the Jordan Music Centre of Tsang Fook Piano in Kowloon, Hong Kong; the interview with Green was conducted at the Millennium Music Software store in Nottingham, UK. The duration of the interviews ranged from 30 to 45 minutes. After they had agreed to participate, the interviews were arranged at a suitable time and place. Each interview commenced with a brief introduction outlining the purpose of the study and permission was requested to audio record the interview.

The interviews were transcribed from the audio recordings and double checked by the researcher. An email with the interview script attached was

sent to each participant after the transcriptions were completed, thanking them for their participation and asking them to confirm the validity of the interview transcripts. They were also asked to consent to their identity being revealed in this thesis. All of the participants confirmed the interview transcripts and agreed to their identity being revealed.

5.3.3 Data Analysis

O pen coding (Charmaz, 2004) was used to analyse the data in this study. The researcher read all of the interview transcripts and compiled a set of recurring 'themes' in the reports. Themes were given titles such as 'relationship and interactions with music teachers' and 'music software business', which were then used to categorise relevant statements for the data collection. To provide an example, a statement from a participant is analysed below:

> *...Compared to the retail market, the education market is much larger, particularly software such as music notation software. The second reason is that only schools are eligible to purchase a volume license. Groups of random customers are not authorised to purchase a volume license. A school license has restrictions. (Lui)*

Since Lui's comment is about music software retailers' reasons of targeting music teachers as their customers, it is categorised under the theme of 'relationship and interactions with music teachers'. By considering interview findings related to interactions and relationships within and

between the three ecologies, the dynamics associated with the ecology of software business were identified and added or amended to the preliminary model of ecology of software busniess as shown in Figure 9. Detail is described in Section 5.6.

5.3.4 Interview Questions

The questions were grouped into four categories with reference to the focus questions: (1) characteristics of the music software business; (2) relationships with music software developers and music teachers; (3) knowledge transfer between music software developers and music teachers; and (4) perspectives towards music software.

The music software retailers were asked the following questions.

Characteristics of the music software business

- What music software does your shop sell?
- What are the most popular types of music software?
- Who are the major competitors in the music software business?
- What discount do you offer to music teachers and students when purchasing music software?

Relationships with music software developers and music teachers

- What questions do those customers ask when they purchase the software?

- What are the customers' key concerns and criteria when purchasing music software?
- What problems do those customers have after they have purchased the software?
- How do you receive feedback from customers?
- How do you provide feedback to software developers?

Knowledge transfer between music software developers and music teachers

- What kinds of information do you provide to music software developers concerning their software development process?
- What training and workshops does your business provide?

Perspectives towards music software

- What are the recent trends in the music software business?

5.4 Findings

The data from the semi-structured interviews were categorised into four themes:

1. The music software business
2. Relationships and interactions with music teachers
3. Relationship and interactions with music software developers
4. Perspectives on music software in music education

Theme 1. The music software business

Millennium Music Software is a music technology specialist retailer that sells music hardware, electric instruments, software packages and studio equipment. Tsang Fook Piano is an all-round music centre providing and selling music services and products, ranging from acoustic and electric instruments, stage and studio audio equipment, music software and hardware packages, music accessories and scores. Both retailers are the leading music software sellers in their regions.

As the leading music technology specialist in the UK, Millennium Music Software covers a large market and sells more music technology products than Tsang Fook Piano. Major music software titles can be found in both retailers' software product lists, such as *Sibelius*, *Pro Tools*, *Cubase*, *Sonar*, *Ableton Live*, *CakeWalk*, *Band In A Box* and *Auralia*. Minor titles such as digital audio workstation plug-ins and software synthesiser patches can be found in Millennium Music Software. Both retailers offer standard and educational versions of the music software they sell. Both representatives said that there is no difference between the standard and educational versions of the music software, except that the educational versions are discounted.

The retailers' customer bases range from those who know nothing or little about music technology to music professionals. Customers' interests range from using music software for fun to earning a living.

Our customers range from knowing nothing about music technology to music professionals. Their interests range from using music software for fun to earning a living. (Liu)

Everyone, from playing music for interest, to students, teachers and music production professionals. (Green)

Both representatives reflected the current trends in the music software business. They pointed out that other than the competition between retailers, the online market had gradually become the largest competitor in the music software business. As the Internet becomes more popular, customers may switch to the online market to find music software. They monitor the software prices from online stores to make sure their prices are competitive.

...the main competitors are actually the online music shops that sell the same things as us. (Green)

(The major competitor is) Tom Lee. We do not have many competitors in the local market. Actually, our largest competitor is the online market. The Hong Kong local market is small and as the Internet grows more popular, customers may switch to the online market to search for music software. We always monitor the software prices from online stores to make sure our selling prices are competitive enough. (Liu)

More and more software developers have begun to offer download versions of their software products in recent years. The boxed, 'physical' versions of music software packages are getting smaller and smaller, and the menus are starting to disappear from the software packages, which customers can download from the Internet. If the file size of the software is not too large, music software retailers will gradually offer download versions on behalf of

the software developers. The main advantage of offering download versions is the shortened trade duration, as retailers can get a download to the customer a lot faster than a boxed version: downloads only need a serial number, which can be provided by email. Moreover, a download version is normally cheaper than a boxed version because of the reduced transportation cost. The trend is for music software retailers to sell more and more download versions, which customers find increasingly acceptable. Retailers can offer instant delivery of download versions, and can stock fewer boxed versions to save physical space and avoid unsold, outdated software.

Theme 2. Relationships and interactions with music teachers

Music teachers form a large part of the customer base for both music software retailers. Compared to the retail market, the education market is much larger for certain software, such as music notation software. As the people in charge of purchasing music software for a school or institute, teachers are the key customers for volume licenses (i.e., multi-use software licenses). They also train students to use the music software, who then become future consumers of music software products.

> *The second reason is that only schools are eligible to purchase volume licenses. Groups of random customers are not authorised to purchase volume licenses. A school license has restrictions. We do pay a lot of attention to the education market. The market private or professional user market is too small, especially for holding workshops for them. (Lui)*

When purchasing music software, music teachers are mainly concerned about its major functions: they need to make sure the software suits their purpose. Price is another concern, especially when purchasing a volume license. They also consider the ease of use of the software, and compare it with other music software brands that serve the same purpose. They will search for reviews of the music software on the Internet, or download the demo to compare different software brands.

> *...the price, what sorts of things they are looking for, and making sure that they get the right product. (Green)*

> *First of all they take the purpose it's to be used for into consideration. Then they will consider the ease of use compared with other music software that serves the same purpose. They will also search the software reviews on the Internet, or download the demo to try and compare. (Lui)*

The problems that music teachers encounter in using music software tend to be technical problems. For example, they may not know how to install the software, or encounter system errors after installation, or don't know how to connect the software to hardware. The music software retailers have technical staff to answer such questions. If the technical problem cannot be solved immediately, or if there are problems such as registration problems that cannot be answered by the music software retailer, then the technical staff will forward the problem to the software developers. The two retailers usually receive feedback from customers by email or in person.

Both music software retailers have relationships with schools in the nearby region. Millennium Music Software has a lot of cooperation with the Confetti Institute of Creative Technologies in Nottingham, one of the UK's biggest audio engineering schools. Tsang Fook Piano holds a lot of workshops to teach music teachers how to use particular brands of music software. These workshops focus on the applications of the software in an educational setting. For example, in workshops on music notation software the focus may be on how to use the software rather than on its sound library feature.

> *We do quite a lot of cooperation with the Confetti Institute of Creative Technologies, one of the UK's largest audio engineering schools. (Green)*

> *(The workshops) mainly target schools. Schools always buy a volume license for software – maybe 40 at a time, or half a computer room, 20. Mostly music teachers and IT teachers. IT teachers usually support music teachers on technical issues, that's why we also target IT teachers. (Lui)*

Theme 3. Relationships and interactions with music software developers

The two representatives said that they provide customer feedback to the developers through the Internet. Tsang Fook Piano also provides feedback through music technology tradeshows such as LimeZone in the US or other tradeshows in Frankfurt and Beijing. They also receive updates from software developers at tradeshows, and exchange information about software sales, marketing and sales strategies.

Mostly through email, and sometimes at tradeshows - LimeZone in the US, or other tradeshows in Frankfurt, Beijing. (Lui)

We speak through the Web, to see how they can help us to sell their products. (Green)

Software developers rely on local distributors – music software retailers or sizable schools in other countries – to hold workshops or seminars on their software products. Software developers may provide cash sponsorship or souvenirs as incentives to retailers to hold workshops and seminars.

Software developers seldom hold workshops themselves. They will rely on the local distributors, or sizable schools in other countries. They will actively request us to hold workshops or seminars, maybe once a year. Some of them will have cash sponsorship or souvenirs. (Lui)

Theme 4. Perspective on music software in music education

Lui pointed out that music notation software and digital audio workstations are the most popular types of software from the perspective of music teachers. Other types of software, such as digital audio workstation plug-ins and sound libraries, appeal more to professional users such as music producers.

Two main types – music notation software, e.g. Finale and Sibelius, and digital audio workstations, e.g. Pro Tools, Cubase, Logic, Sonar and Digital Performer. Other types include plug-ins, e.g. sound libraries. Plug-ins are more appealing to professional users such as music producers. (Lui)

The representatives also provided their views on the rapid software updates and releases of new software versions in the current music software development market. They said that some music teachers find the new functions in updated versions useful. However, some functions cause trouble for users. For example, an old version of *Finale* cannot open the score files generated by newer versions. *Sibelius* is more flexible in such respects. *Finale* tries to force users to purchase the most up-to-date version of the software as the company's business strategy. Most of the software companies offer discounts for updating their software products.

5.5 Discussion

Both music software retailers reflected that technological advancements affect the way the music software business operates, from how software products are produced to how they are delivered to customers (e.g. Keeling, Keeling & McGoldrick, 2011; Liao & Shi, 2009; Manyika, Roberts, & Sprague, 2007; Padgett & Mulvey, 2007). Technological advances mean that software no longer relies on the retail business and virtual versions of software products are replacing the physical versions as the means of delivery.

5.5.1 Product Delivery Practice

The interviews revealed that download (virtual) versions of software products are increasingly replacing boxed (physical) versions. From the perspective of music software retailers, this trend not only increases the flexibility of product delivery, but also reduces the need for physical storage. They no longer need to stock physical software packages, and there is no longer a problem with software products being out of stock.

More and more music software developers are establishing direct transaction channels with their targeted customers instead of going through software retailers, reflecting the implementation of the standard software B2C model described by Valtakosi and Rönkkö (2009). Customers find it easy to access a software developer's official website and download the full version of the software product via electronic payment and high-speed Internet data transfer, or via the App store on the iPhone mobile platform and Google Play on the Android mobile platform. Music software retailers have no role to play in such a trading environment, so the online business not only benefits the music software retailing business, but also creates a threat to it.

Music software retailers can no longer rely on business through the physical delivery of products, but must also use other channels such as online business, sales campaigns and training workshops. Retailers must increase their customer base to survive in the competitive music software market. The provision of alternative product delivery methods is one practice that

contributes to the commercialisation of software in the ecology of the music software business.

5.5.2 Marketing Practices

The interview data and the advertisements on the participants' company websites indicated that the companies' marketing strategies tend to focus on product features or highlights aimed at professional users of the product. New and advanced functions of the software are emphasised, but not how they are associated with the users' experiences – i.e., how the software could facilitate the teaching and learning of music in the context of music education.

'New users need to hear stories they can relate to, not just stories about successful users' (Lenberg, 2010, p. 67). As the focus of music making and enjoyment is shifting from professional users to everyday consumers, the marketing strategy adopted in the music software business should consider the novice market in addition to the professional market. Marketing materials should not only provide knowledge of the software's functions, but should also illustrate how the software can help to solve their problems, such as how music software can facilitate music teaching and learning in the music classroom.

5.5.3 Glocalisation Practice

The interview data revealed that the music software business in different regions follows a similar business pattern, and faces the same problems, threats and favourable circumstances. The globalised economy has brought software businesses together to face the same challenges and opportunities, forcing participants to change, evolve and innovate within and beyond the music software market.

Successful global companies such as McDonald's and HSBC are famous for their glocalisation strategies, which market their software products globally with an emphasis on local conditions in different regions (Han, 2008; Koller, 2007). In the music education context, the music curriculum differs from one to another country, as does the level of technological involvement in education and teachers' computer proficiency.

As software becomes more customised and personalised, music software should be ready to serve local and global needs at the same time. Most music software developed in recent decades has implemented internationalisation and localisation by adapting the software to the different languages, regional differences and technical requirements of targeted regions. Applying a glocalisation strategy in the music software business relies on music software retailers' role in glocalising software products within different regional markets. This could include bridging the gap between the technical knowledge needed to handle the software and

customers' technological proficiency, or adjusting the pricing to meet the affordability considerations of local schools.

5.5.4 Version Control

The interview data revealed the rapid updating of software and releasing of new versions in the current music software market. Music software developers try to develop new functions to increase the competiveness of their products. However, adding new functions that do not fit the needs of end users can have adverse effects.

Although the development of new functions relies on the accuracy of the software developers' requirement engineering, they also need to be aware of the effect of version control. Frequent releases of new software versions with attractive new functions can boost sales volume, but can have the opposite effect if the new version is not attractive to the users. This could result in customers either shifting to other software brands or becoming more reluctant to upgrade. How music software developers control the release of software versions is thus a dynamic between the developers and the commercialisation of the software.

5.5.5 Users' Feedback

The dynamics of users' feedback were identified in the study of the ecology of software development reported in the previous chapter. Although the communication channel for feedback was not identified in the previous study, this study revealed that one of the music software retailers' roles is to act as a communication channel for the end users to provide feedback to the music software developers. This requires not only skills and labour, but also a suitable mechanism for communicating the users' feedback to the developers.

5.5.6 Software Training and User Support

The interview data showed that music software retailers have mechanisms for addressing user support needs, such as cooperating with local schools and holding training workshops. Research has shown that music teachers have little confidence in their ability to integrate technology effectively into their lessons (Taylor, 2003), and music software is often beyond the technical expertise of a school's technology expert (Noxon, 2003). Thus, software training and after-sales support is important to the software retailing business as the complexity of software technology and the technical knowledge required to use it increase.

5.6 Summary

The findings of this study were applied to the preliminary ecology model in Section 2.5 (see Figure 10). The refined model captures the dynamics of the ecology of the music software business revealed in the interview data. The refined ecology of music software business is presented in Figure 14.

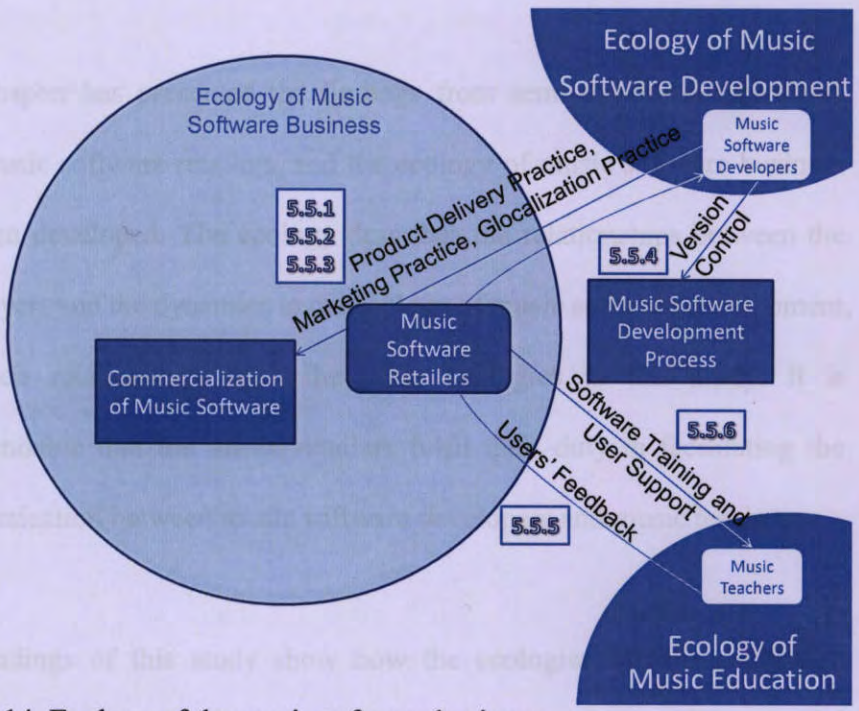

Figure 14. Ecology of the music software business

The ecology of the music software business is shown in Figure 14. It describes the relationships between the key players and the dynamics of the ecology of the music software business, and their relationships with the other ecologies in this study. The numbers in Figure 14 correspond to the relevant sub-sections in the Section 5.5.

The music software market has grown incredibly fast and has become a highly competitive global market (Largillier, 2007; Maedche et al., 2012). The same music software products are being sold at very similar prices by retailers around the world. As such, the findings of this study should be applicable to countries beyond Hong Kong and the UK.

This chapter has presented the findings from semi-structured interviews with music software retailers, and the ecology of music software business has been developed. The ecology describes the relationships between the key players and the dynamics in the ecology of music software development, and their relationships with the other ecologies in this study. It is commendable that the music retailers fulfil their duty in facilitating the communication between music software developers and music teachers.

The findings of this study show how the ecologies of music software development and the music software business interconnect with each other. The next chapter shows how the ecology of music education links with these two ecologies.

CHAPTER 6

Study 3: Ecology of Music Education

This chapter describes the planning and administration of the data collection process, and reports on the findings of this study specific to the ecology of music education. The chapter begins with an overview of the ecology of music education, followed by the focus questions which guide the implementation of questionnaire survey and semi-structured interviews. The next section describes the research design of this study followed with the findings, discussion, and concludes with a summary.

6.1 Overview of the Ecology

Music education is one of the three ecologies examined in this study. The ecology of music education is broad and consists of practices ranging from individual learning of instruments to music therapy and digital music-making. This thesis focuses on the usage of music software in the classroom context by music teachers to facilitate their teaching. Sections 2.6 and 2.7 have reviewed the related literature, and a preliminary model of the ecology of music education was presented in Section 2.8 (see Figure 11).

The literature reviewed did not reveal the interconnections between the ecology of music education and the other two ecologies. It was not able to

clarify how music teachers choose and apply music software in their teaching, as well as their relationships with music software retailers and music software developers.

6.2 Focus Questions for the Ecology of Music Education

The main purpose of the study is to uncover more details about the missing but needed information concerning the ecology of music education. Based on literature review, the following focus questions have been developed to guide this study:

- What are the music teachers' criteria for selecting the music software to be used for teaching?
- How do music teachers use music software to facilitate their classroom-based music teaching?
- What are the most desirable functions of music software, from the music teachers' perspectives?

6.3 Research Design

Two data collection methods were used in this study: a questionnaire survey and semi-structured interviews with 15 music teachers from Hong Kong (see Section 6.3.2).

6.3.1 Context of Hong Kong Education Policy on ICT and Music Education

After the transfer of Hong Kong's sovereignty from the UK to China, the government of Hong Kong adopted a 'biliterate and trilingual' language education policy. Under this policy, both Chinese and English are acknowledged as official languages, with Cantonese being acknowledged as the *de facto* official spoken variety of Chinese in Hong Kong. The latest medium of instruction (MOI) policy allows primary and secondary schools to select either Cantonese or English as the MOI (Evans, 2011).

Since the launch of *Information Technology for Learning in a New Era: Five-Year Strategy 1998/99 to 2002/03* – the five-year strategic plan to promote the use of information technology to enhance teaching and learning (Education and Manpower Bureau, 1998a, 1998b) – significant progress has been achieved in improving the technological infrastructure, teacher training and curriculum support available in Hong Kong (Education and Manpower Bureau, 2005).

The *Music Curriculum Guide* for primary and junior secondary schools was published in 2003 and implemented in 2006, followed by the *Music Curriculum and Assessment Guide* for senior secondary schools, published in 2007 and implemented in 2009. Both curriculum guides state the same four key learning targets: (1) developing creativity imagination; (2)

developing music skills and processes; (3) Cultivating critical responses in music; and (4) understanding music in context. Both curriculum guides also emphasise role of technology in music curricula (HKCDC, 2003; HKCDC & HKEAA, 2007).

Leung (2003) has criticised the emphasis on teacher-centred approaches in the schools' music curriculums, which fail to acknowledge the current trend for student-centred and computer-assisted music learning. A later study (Lee, 2010) revealed that the provision of music-orientated IT training for teachers was adequate in terms of quality and content, but that the teachers were unlikely to attain a level of competency in music technology that was sufficient to meet their needs when teaching music.

6.3.2 Participants

Fifteen music teachers – five males and ten females – were purposively selected and invited to participate in the questionnaire and individual interviews. Six of them taught in primary schools, and nine taught in secondary schools. Out of the nine secondary school teachers, eight of them were teaching the Hong Kong local curriculum whilst the ninth was teaching the International Baccalaureate (IB) programme. They were all recent music-education graduates (two of them were also pursuing postgraduate studies at the master's level) from the Hong Kong Institute of Education, who were selected for their experience with the most recent

teacher education curriculum in Hong Kong. The Institute is one of eight subsidised tertiary institutes under the UGC of Hong Kong and is the only one dedicated to teacher education. It offers Bachelor of Education (Music) and Post Graduate Diploma in Education (Music) since 1998, nurturing the majority of music teachers in Hong Kong.

Table 4 shows the participants' self-rated computer proficiency and confidence in using music software for teaching purposes, as reported on the questionnaire.

Table 4

Participants' self-rated computer proficiency and confidence in using music software for teaching purposes

Participants' self-rating (5=highest)	**Computer proficiency (n=15, mean=3.40)**		**Confidence in using music software for teaching purposes (n=15, mean=4.07)**	
	Pri (n=6)	Sec (n=9)	Pri (n=6)	Sec (n=9)
5	1	0	1	1
4	2	3	2	3
3	3	5	3	3
2	0	1	0	2
1	0	0	0	0

The distribution of ratings shows that the participants generally rated themselves moderately competent both with general computer proficiency and with the use of specific music software for teaching purposes, with the

latter being rated slightly higher than the former. Primary school music teachers rated themselves slightly higher than secondary school music teachers, on average.

6.3.3 Administration of the Survey and Interviews

The data were gathered via a questionnaire and a number of semi-structured interviews, both conducted in English. Although English was not the first language of all the participants in this study, they all possessed sufficient proficiency to complete both the questionnaire and the interview. The first part of the questionnaire asked for the demographic information of the participants, including gender, education level and the type of school they worked in, before moving on to ask for self-rated computer competence and confidence in using music software for teaching purposes. The second part of the questionnaire gathered quantitative data concerning types of music software used by participants, their sources of knowledge about music software, and the types of music activities that they thought were suitable for incorporating music software into. The questionnaire can be found in Appendix A.

The data collection was implemented in the summer of 2011. A formal email was sent to each participant to seek their participation in the study, and to explain the purpose of the study and the data collection process. Each of them had to participate in one data collection session, consisting of both

the questionnaire and the interview. The sessions were undertaken in person in Hong Kong, and each session lasted between 15 and 30 minutes. A suitable time and place was arranged after receiving their agreement via email. Each session commenced with a brief introduction detailing the purpose of the study, and permission to record the interview was secured.

The interviews were transcribed from the audio recordings and double-checked by the researcher. An email with the interview transcript attached was then sent to each interviewee, thanking them for their participation and asking them to confirm the validity of the transcript. All of the interviewees confirmed the interview transcripts. The real names of the interviewees are not disclosed here for anonymity reasons.

6.3.4 Data Analysis

Open coding (Charmaz, 2004) was used to analyse the data in this study. The researcher read all of the interview transcripts and compiled a set of recurring 'themes' in the reports. Themes were given titles such as 'desirable functions' and 'pedagogical needs', which were then used to categorise relevant statements for the data collection. To provide an example, a statement from a participant is analysed below:

> *I would prefer an automatic translating function that can translate sheet music to an editable computer file. I know there is such a*

technology available, but it is not accurate enough for my needs. (Millie)

As this participant's comment refers to a particular function that she wants to use, it was categorised under the theme of 'desirable functions'. By considering interview findings related to interactions and relationships within the ecology of music education alongside considerations of links to the other two ecologies, the dynamics associated with the ecology of music education were identified and added to the preliminary model of the ecology of music education (Figure 10). The updated model is presented in Section 6.6 (Figure 16).

6.3.5 Interview Questions

The questions were grouped into four categories with reference to the focus questions: (1) criteria for the choice of software; (2) applications of music software in classroom teaching; and (3) software functions desired by music teachers.

Music teachers were asked the following questions:

Criteria for choice of software

- Which particular brands of music software have you used?
- If you have tried other brands of the same kind of software, which do you think is better and why?

Applications of music software in classroom teaching

- How is the software used (e.g. demonstration during the class, preparation of teaching aids)?
- Which software functions have you used from each type of software?

Software functions desired by music teachers

- What additional functions do you wish the software had?

6.4 Findings

In this section, data from the questionnaire and semi-structured interviews are reported under five themes:

1. Type of software used
2. Criteria for choice of software brands
3. Technology-integrated classroom music activities
4. Applications of music software for teaching purposes
5. Software functions desired by music teachers

Theme 1. Type of software used

Table 5 shows the types of music software used by participants for teaching purposes.

Table 5

Types of software used by participants

Type of Software Used	Response (n=15)	
	Pri (n=6)	Sec (n=9)
Music Notation Software	7	7
Mobile apps	4	4
Audio editing and recording software	0	4
Theory and aural training software	2	1
Audio loop composing software	0	2
Music sequencing and recording software	0	1
Auto arranging and accompaniment software	0	1
Elementary music teaching software	1	0
Step sequencing software	0	0
Other	0	1 (SmartMusic)

All but one participant used music notation software, with mobile apps as the second most used software. Audio editing and recording software, theory and aural training software and audio loop composing software were used less frequently, and step-sequencing software was not used by any of the participants. Software for audio editing, recording, audio loop composing, sequencing, auto-arranging and accompaniment was not used by any of the primary school music teachers. The music teacher from the international school was the only one who used *SmartMusic* – software for assessing vocal/instrumental performance skills.

The interview data showed that teachers with higher self-rated computer proficiency and confidence with using music software for teaching purposes

used a wider range of software in their teaching. For example, a participant who rated both 5 for computer proficiency and confidence in using music software said that he used music notation software, audio editing software and aural training software in his teaching. Another participant who rated 5 for computer proficiency and 4 for confidence in using music software used music notation software, music sequencing software, audio editing software, theory and aural training software, audio loop composing, auto-accompaniment software and software for assessing vocal/instrumental performance skills that requires both musical and computational understanding to handle the software.

Theme 2. Criteria for choice of software brands

The survey data showed that 11 of the participants acquired their knowledge about music software through university coursework/study, 10 via the Internet, 6 from friends and 2 from colleagues. None of the participants indicated the source of their knowledge was music software retailers. The interview data revealed that most of the participants only learnt about and used one particular brand of software for each type of software, and tended to continue to use that brand in their teaching. They learnt about other brands of software through the Internet, friends and colleagues. The interview data revealed that their criteria for selecting the brand of software used included: (1) availability of software in their school; (2) purpose; (3) ease of use; (4) familiarity; (5) range of available functions; (6) quality of output generated by the software (e.g. audio playback, music scores editing);

and (7) whether the students could use the same software at home. Two statements by participants illustrate this:

> *I have tried Overture and Finale for music notation software. Since the computers in my school do not come with Finale, I use Overture most of the time as it is free and I can install it everywhere. My tasks are simple, so Overture is sufficient for my needs. (Tony, primary school music teacher)*

> *I use Finale and Overture for score editing...most of the time I use Finale since its functions are more comprehensive and it can do everything I require. (Stella, primary school music teacher)*

As most of the participants in this study did not have access to all of the software that they would like to use at school, they had to search for alternatives, or purchase the software themselves. Due to tight budgets in music departments, none of the participants would consider purchasing the software through their schools, and most did not have the authority to purchase new items for their schools anyway, due to their junior status.

Theme 3. Technology-integrated classroom music activities

Table 6 shows the participants' ratings of music activities that they considered suitable for incorporating music software.

Table 6

Music activities suitable for incorporating music software

Music activities suitable for incorporating music software	**Response (n=15)**	
	Pri (n=6)	Sec (n=9)
Composition	6	6
Music theory	3	5
Aural skills training	4	4
Music appreciation	4	2
Instrument/vocal training	0	4
Music history	0	0
Other(s)	0	0

The survey data showed that the most popular music activity to incorporate music software into was composition, followed by music theory, aural skills training, music appreciation and instrument/vocal training. None of the primary school teachers thought that instrument/vocal training could incorporate music software. Participants rated music theory and aural skills training as the second most suitable activity for incorporating music software.

Theme 4. Applications of music software for teaching purposes

The interview data indicated that the music teachers mainly applied music software in their teaching in three ways: (1) preparation of teaching aids; (2) demonstration of music (e.g. timbral differences of different instruments, rhythmic patterns) and (3) task-orientated activities.

All participants reported that they used music software to prepare teaching aids, for example using music notation software to insert musical examples into worksheets, exercises and examination scripts.

> *I use Sibelius to make music scores and playback the music to students. Sometimes the music textbook packages do not provide the single melody extracts on the CD samples, so I have to make them myself. (Kobe, primary school music teacher)*

Four participants used notation software and mobile apps to demonstrate how particular instruments or melodies sounded.

> *I use Sibelius to demonstrate the timbre of particular instruments, too, such as the snare drum, which is better than describing them verbally. (Kobe, primary school music teacher)*

Both the secondary school teachers and the international school teacher utilised music software for task-orientated activities. Various types of music software were used for different tasks, for example, the participant who used *SmartMusic* to assess students' instrumental performances commented on the design of the software in this way:

> *If you play something wrong, it [SmartMusic] will tell you. But it does not assess how long the notes you play are. It only assesses the first beat of any note on the score. (Sharon, international school music teacher)*

Another participant taught students to use *Audacity* (audio editing software) to make music-minus-one audio files.

Theme 5. Software functions desired by music teachers

The interview data revealed that there were two types of desirable functions: those currently unavailable in the software; and those functions that are available but not used by music teachers. Some unavailable functions included:

> *I would like the music notation software to automatically generate [solfeggio] from the melody on the score by one simple click. This would be very helpful for students to learn singing. (Joyce, primary school music teacher)*

> *When I use Finale to make sheet music, the layout is not what I want it to be. I want to generate one page of sheet music so I can distribute it easily to my class. But Finale always manages the score layout based on aesthetics rather than number of pages, [for example] generating one and a half pages of sheet music. (Victoria, primary school music teacher)*

> *I want software that can notate the melody that I sing through the microphone. Sometimes students can sing the melody but have no idea how to notate it. Such a function could help. (Galilee, secondary school music teacher)*

This last function has yet to be developed in currently available music notation software, but it is available in some audio editing and recording software such as *Digital Performer*. There were two main reasons given for music teachers not using the available functions supposed to meet these

needs. The first was that teachers felt the available functions did not perform as well as what they would expect:

> *I would prefer an automatic translating function that can translate sheet music to an editable computer file. I know there is such technology available, but it is not accurate enough for my needs. (Millie, primary school music teacher)*

The second reason given was the coverage limitations in currently available music software:

> *I would like to see sound libraries for Chinese instruments in music notation software. (Kobe, primary school music teacher)*

> *It would be great if the software for performance assessment and evaluation can let the students know how to improve their skills instead of only detecting errors. (Sharon, international school music teacher)*

A summary of the survey data and types of software brands used by participants for teaching purposes from interview data are presented in Tables 7 and 8. Participants' self-rated computer proficiency and confidence in using music software for teaching purposes are included in Table 8.

Table 7

Summary of survey data

	Pseudo names	Self-rated computer proficiency	Self-rated confidence in using music software for teaching purposes	Type of software used	Source of knowledge about music software	Music activities suitable for incorporating music software
Primary school music teachers	Millie	4	4	Music notation	University coursework/study	Composition
	Victoria	3	3	Music notation Mobile apps	University coursework/study	Music theory Composition
	Cindy	3	3	Music notation	University coursework/study Internet	Music appreciation
	Kobe	5	5	Music notation Theory & aural training Elementary music teaching software Mobile apps	Internet	Music theory Composition Aural training Music appreciation
	Stella	3	3	Music notation Theory & aural training Mobile apps	Internet Friends	Music theory Composition Aural training
	Ada	4	4	Music notation Mobile apps	University coursework/study Internet	Composition Aural training Music appreciation
Secondary school music teachers	Joyce	4	4	Music notation Audio editing & recording Audio loop composing Mobile apps	Internet	Music Theory Composition
	Suki	4	4	Music notation	University coursework/study Colleagues	Composition Aural training

					Music appreciation
	Galilee	3	2	Audio editing & recording Mobile apps	Friends	Music appreciation Instrument/vocal training
	Tony	3	3	Music notation	University coursework/study	Music theory
	Phoebe	3	3	Music notation Mobile apps	University coursework/study Internet Friends	Music theory Composition Aural training Instrument/vocal training
	Esther	2	3	Music notation Mobile apps	University coursework/study Internet Friends	Composition Instrument/vocal training
	Alice	3	4	Music notation Audio editing & recording	University coursework/study Internet Friends	Music theory Composition Aural training Music appreciation
	Mandy	3	2	Music notation	University coursework/study Internet	Composition Aural training
International (secondary) school music teacher	Sharon	4	5	Music notation Music sequencing & recording Audio editing & recording Theory & aural training Audio loop composing Auto arranging & accompaniment Other (SmartMusic)	University coursework/study Internet Colleagues Friends	Music theory Composition Aural training Instrument/vocal training

Table 8

Software brands used for teaching purposes (* denotes free/open-source software)

Pseudo names	Self-rated computer proficiency	Self-rated confidence in using music software for teaching purposes	Software brands used for teaching purposes
Millie	4	4	Finale
Victoria	3	3	Finale, virtual piano* (mobile apps)
Cindy	3	3	Finale
Kobe	5	5	Sibelius, Finale Notepad*, Music Ace I/II, Musition, Auralia
Stella	3	3	Overture*, Finale, virtual piano* (mobile apps), unknown aural training software
Ada	4	4	Overture*, Finale, virtual piano* (mobile apps)
Joyce	4	4	Sibelius, Audacity*, GarageBand, virtual instruments* (mobile apps)
Suki	4	4	Sibelius, Finale
Galilee	3	2	Audition
Tony	3	3	Overture*, Finale
Phoebe	3	3	Overture*, Finale, virtual piano* (mobile apps)
Esther	2	3	Sibelius, Finale, metronome and virtual piano* (mobile apps)
Alice	3	4	Sibelius, Overture*, Audacity*
Mandy	3	2	Sibelius, Noteflight*
Sharon	4	5	Sibelius, Finale, SmartMusic, Audacity, GarageBand, Logic

6.5 Discussion

The survey and interview data showed that composition was the most common classroom activity that involved music software, and that music notation software was the most popular software to be incorporated in the composition teaching process. This echoed previous studies that showed software technology could benefit students learning composition (Airy & Parr, 2001; Chen, 2012; Gall & Breeze, 2005; Manzolli & Verschure, 2005).

The summary of the survey data in Table 7 shows that primary and secondary school music teachers rated similarly in their self-rated computer proficiency and confidence in using software for teaching purposes, but had different applications for, and thoughts on, the incorporation of software into classroom activities. Primary school teachers did not use music software for advanced music activities such as music arranging, recording and composing, and none of them thought that instrumental/vocal audio could incorporate music software. This is perhaps due to differences in music activities between primary and secondary schools.

6.5.1 Competence for Teaching with Technology

Interview data revealed that content knowledge and technological competence were two important factors that determined music teachers' applications of music software in their teaching.

Most of the participants had only ever been taught how to use one particular brand per type of music software, and they all continued to use that brand in their teaching. This indicated that the content of music teacher training actually influenced how music teachers apply music software in their teaching. Additionally, teachers' lack of exposure to multiple brands of software limited their ability to select the appropriate music software for their needs. Teacher training should not only provide knowledge of music software, but also the skills in how to apply that software in schools. This influences how teachers apply software in their teaching, and thus what students learn from computer-assisted music lessons.

Table 8 shows that participants with higher self-rated computer proficiency and confidence in using music software for teaching purposes used more music software than those who rated themselves lower (e.g. Kobe, Joyce, and Sharon). The literature confirms that technological competence is one of the factors that affects music teachers' applications of music technology in their teaching (Basen-Smith, 1999; Gall, 2013). High technological competence enables teachers to solve both their own and students' problems concerning the use of music software, which minimises the difficulty of integrating music software into teaching practice.

Other than content knowledge and technological competence, other factors effecting software usage in the literature include the skills to integrate music technology into music teaching; and teacher attitude towards the usefulness of technology in music teaching (e.g. Busen-Simth, 1999; Bauer, Reese & McAllister, 2003; Baylor & Ritchie, 2002; Dorfman, 2008; Gall, 2013; Gall et al., 2012; Greher, 2011; Ho, 2007, 2009; Perkmen & Cevik, 2010; Roblyer, 2006; Welch, Purves, Hargreaves & Marshall, 2011). The emphasis here is the influence of IT-specific teacher training on music teachers' competence with using technology to teach. As Lee (2010) has mentioned, the provision of music-orientated IT training for teachers in Hong Kong was adequate in terms of quality and content, so more recently graduated music teachers should have the basic competence to be able to apply music software in their teaching. Initial teacher training, however, does not have sufficient time to provide comprehensive training on different types and brands of music software. This suggests that professional development is needed to keep music teachers up to date with the latest music technology and relevant pedagogical skills (Bauer et al., 2003).

6.5.2 Choice of Music Software

As reported in Theme 2 of this study, seven criteria were used by music teachers to determine the choice of music software applied in their teaching. These can be classified around three factors: (1) school factors – the

availability of software and the support provided for integrating technology into teaching in their schools; (2) human factors – purpose of software usage, familiarity with the software and availability of software to students; and (3) software factors – ease of use, comprehensiveness of the functions provided and quality of output. Music teachers do not have control over these factors, but they are the gatekeepers to ensuring the most suitable music software is used in their classrooms.

6.5.3 Facilities and Support

Participants reported that the availability of facilities and support in their schools were important factors affecting their applications of music software in the classroom. This included: the availability of computers and other technological equipment in the classroom; the budget allocated to the music department; and the technical support available from the schools' IT staff. Whilst Lee (2010) has pointed out the effects of music teachers' low competency with music technology, the provision of better facilities and support can help to bridge this competency gap and enable these teachers to utilise technology. The findings of this study indicate that the experience of participants did not match Lee's (2010) conclusions that Hong Kong schools' acquisition of IT equipment and music programs' entitlement to shared-access IT facilities for music teaching and learning is 'practically comparable at least to the minimum standards recommended' in countries relatively advanced in educational development (p. 24).

The interview data in this study show that schools did not provide the music software needed by most of the music teachers, forcing them to search for free or open-source alternatives. Although some of the music teachers did have the adequate skills to apply music software in their teaching, the amount the school's budget allocated to software was insufficient to allow music teachers to purchase the music software they needed. There seems to be a disconnection between music teachers' needs and schools' support in terms of music software provision. The lack of availability of computers in the classroom was another hindrance that discouraged music teachers from incorporating music technology into their teaching (Busen-Smith, 1999; Roblyer, 2006).

The previous chapter discussed the software training and support provided by music software retailers. However, when music teachers opt to use free or open-source software, music software retailers have a much more limited role to play, because their business is to sell commercial music software for profit. This could explain why the survey data showed that none of the recently graduated participants' knowledge of music software came from the retailers. So there appears to be another disconnect; this time between music teachers and music software retailers. However, retailers could still have a role in providing training and workshops on new music software to music teachers, even if the teachers are not currently able to purchase the software. This could be considered part of the professional training that

keeps music teachers up to date with the latest music technology. With sufficient knowledge of the latest music software, music teachers would potentially be able to use more software and/or better able to apply software for teaching and learning purposes in the future.

6.5.4 Software Design and Music Teaching

Table 8 shows that other than music notation software, participants only used one brand of each type of music software. Due to time constraints, teacher training may not be able to provide comprehensive training for all types and brands of music software, meaning teachers have to acquire knowledge of other software from friends, colleagues and the Internet. Through those alternative sources of information, they could learn about the abilities, strengths and weaknesses of different brands of software, enabling them to assess the quality of each brand of software according to their specific needs and prioritise brands accordingly.

Some of the music teachers' desired functions, however, were not present in any currently available music software. It is disappointing that music teachers' needs are still not being addressed in the software development process, especially as previous research over a decade ago already pointed out the dearth of interdisciplinary teamwork and user consideration between software developers and music educators (Flores et al., 2001; Krüger et al., 1999; Leong, 2002). Music teachers have diverse expectations of music

software, some of the range reported here included tailor-made interactive options for individual students' instrument learning; designs that facilitate classroom teaching *and* self-directed learning; support for individual music activities; and full coverage of music materials. Software developers need to conduct more research that not only elicits the basic user requirements, but also investigates the music curriculum, teaching process and perspectives of teachers in order to appreciate the full interaction between music teaching and the use of technology. The value of involving music teachers is also discussed in Section 4.5.5, which emphasises the need for better requirement analysis by software developers, as well as better teacher training in software use. As mentioned in the previous chapter, music software retailers can act as the communication channel between music teachers and music software developers to bridge the gap between the two and enable teachers' involvement in the development process. Music software developers, music software retailers and music teachers thus all have to collaborate to achieve best practice in software development, in order to produce better music software for teachers.

Music teachers have their own unique ways of applying music software in their teaching – the use of music notation software to demonstrate the different timbres of different instruments, or prepare music exercise sheets, for example. These practices belong to the teacher-centred approach to music education. Task-orientated activities that emphasised students' self-directed learning were also found in this study (such as music arranging,

recording and composing), however only the secondary school music teachers provided tasks that required advanced music knowledge and skills such as recording techniques and music theory. As the literature suggests, the transformative power of technology is gradually changing the approach to music education; from instructivist (teacher-centred) to constructivist (Beckstead, 2001; Ward, 2009; Wise, 2010; Wise et al., 2011). This change enables students to learn primarily on their own, with teachers acting as facilitators to assist this self-directed learning. For this method of teaching and learning, music software would ideally enable students to learn using the music software as a tool.

An example of this kind of use was discussed earlier, in which one participant required her students to use *SmartMusic* to assess their own performances and make improvements. However, this is the only example in this study that incorporated music software into a constructivist method. The majority of the teachers' applications of music software remained teacher-centred. While there is a lot of learner-centred music software developed (as shown in Section 2.6.2), music teachers have yet to apply this software in their teaching practice. To change this situation, teacher training programmes need to incorporate constructivist methodology in their teaching of pedagogical approaches, i.e. how to encourage students to learn on their own using music technology. This further reinforces the role of teacher training in preparing music teachers for the constructivist music classroom.

The results of this study also highlight the narrow sonic landscape represented by current music software – the 'Western canon' phenomenon described by Webster (2011b). Most music software currently in the market is built based on Western music theories from one thousand years ago. This problem was identified a decade ago when Beckstead (2001) warned music educators to be 'aware of the limitations of such technology [music notation software] and its biases toward discrete, mechanical reproductions and western classical notation systems' (p. 49), and repeated again more recently at the Tanglewood II Symposium 'music educators should be concerned about the effect of technology on the loss of cultural diversity' (Palmer & de Quadros, 2012, p. 19). However, whilst the cultural imperialism of Western classical music still dominates over other cultures, the software development business cannot avoid following this trend. The source of this problem can be traced back to the music curriculum – either in teacher training programmes or in school music education – and the need to find a balance between western classical music and music from the rest of the world.

Discussion on music software design and music teaching revealed that while music teachers' desired functions have not been sufficiently considered in the software development process, the developers' learner-centred emphasis in software is also not used by music teachers. Better collaboration between music teachers and music software developers

is needed to ensure the software developed is highly applicable to music teaching. This also requires the involvement of music software retailers, to facilitate this communication.

6.5.5 Pricing Factor

Participants considered the pricing of commercial music software on three levels: bottom-up, middle-out and top-down (Dede, 1998). At the bottom-up level, they were concerned about whether students could use the software in their home environment. They choose free music software if possible, because students could then download and use the software at home without paying money. At the middle-out level, they would consider purchasing the music software for their own use if they think it is worth it to them to do so. At the top-down level, they would consider whether the allocation of the school budget is sufficient to purchase the music software for the classroom.

Table 8 shows that the music teachers in this study incorporated free or open-source software so that both themselves and their students could always use the software, and there was no risk of a budget problem. Free and open-source software includes *Audacity, Finale Notepad, Noteflight and Overture*. The literature also confirms the prevalence of open-source music software such as *Audacity* and free mobile Apps in school education (Thorgersen, 2010). However, as reflected by participants' concern about

the quality of the output from free and open-source software, the less-structured software development models used by open-source software developers are not able to produce software with sufficiently comprehensive functions for music teachers' needs when compared to the commercial development companies. Due to this, music education in schools still relies heavily on the use of commercial music software, rendering the pricing factor a very important dynamics in determining whether music software is being used in schools. In the study of the ecology of the software business in the previous chapter, measures were highlighted such as the provision of education versions of music software to schools or individual teachers and students at a discounted price. It is gratifying that the music software retailers have made some effort to offer assistance to music teachers who have difficulty purchasing music software.

6.6 Summary

The findings showed that some of the themes in this study connected with those in previous studies – for example, music teachers' feedback on software design in this study echoed with that from music software developers in Chapter 4 (see 6.5.5 for details). In addition to the findings from the studies of the three ecologies supporting each other, the literature review also echoes these findings, resulting in data triangulation as shown in Figure 15, increasing the validity of the results in this thesis.

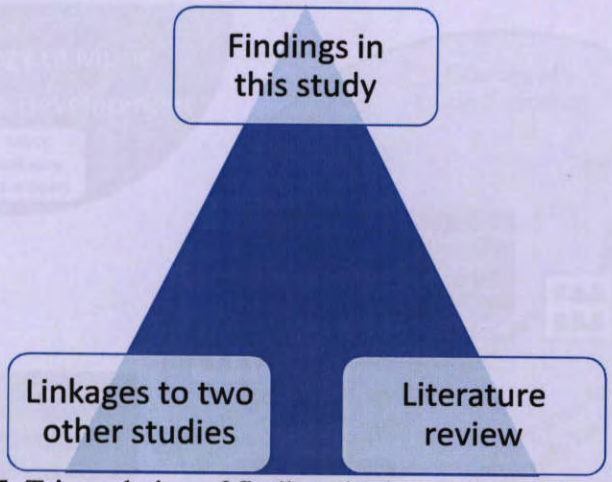

Figure 15. Triangulation of findings in the studies of three ecologies

The findings of this study are supplemented by the findings of previous studies on the ecologies of software development and software business in Chapters 4 and 5. These findings contribute to refining the preliminary ecological model in Section 2.8 (see Figure 11). The refined model captures the dynamics of the ecology of music education as revealed in the interview data. Figure 16 presents the refined ecology of music education.

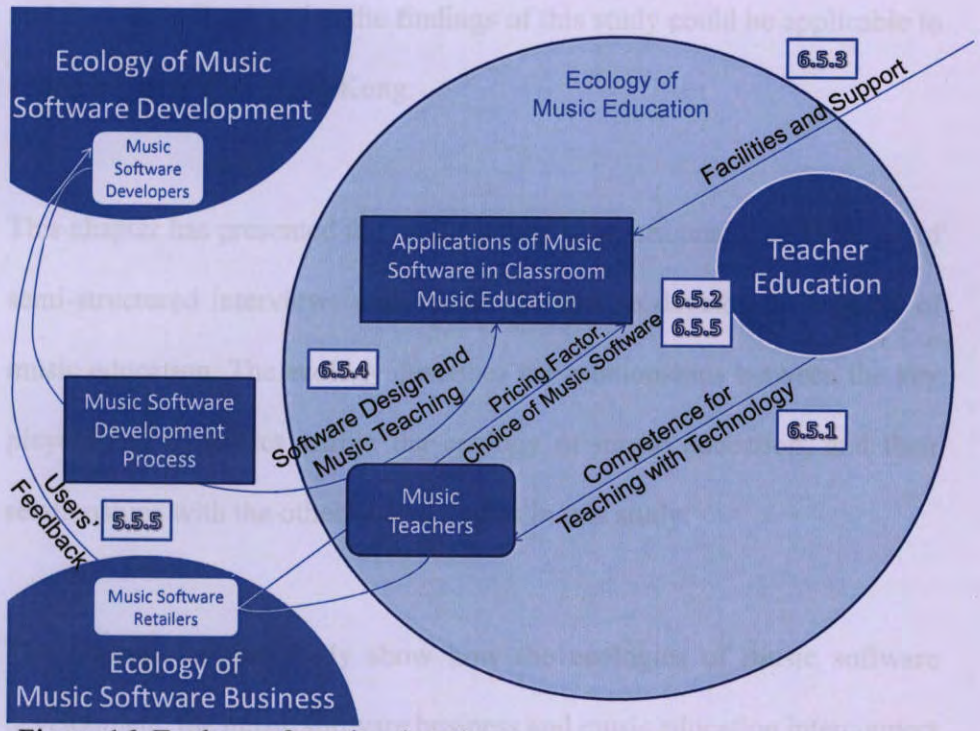

Figure 16. Ecology of music education

The ecology of music education in Figure 16 describes the relationships between the key players and key dynamics within the ecology of music software development, and their relationships with the other ecologies in this study. The numbers in Figure 16 correspond to the relevant sub-sections in the Section 6.5.

The provision of music-orientated IT training for teachers in Hong Kong was adequate in terms of quality and content, where the music curriculum and teaching approaches were designed based on Westernised music and educational theories. The music software used by Hong Kong music teachers came from the global market and was mostly developed in Europe

and the US, indicating that the findings of this study could be applicable to countries other than Hong Kong.

This chapter has presented the findings from a questionnaire and a series of semi-structured interviews with music teachers to develop an ecology of music education. The ecology describes the relationships between the key players and dynamics within the ecology of music education, and their relationships with the other two ecologies in this study.

The findings of this study show how the ecologies of music software development, the music software business and music education interconnect. The next chapter assesses the ways in which these three ecologies interconnect.

CHAPTER 7

Study 4: Music Technology in University Programmes

This chapter describes the planning and administration of the data collection process and reports on the findings of this study specific to music technology in higher education. The chapter begins with the purpose of this study, followed by the focus questions used to guide the semi-structured interviews. The next section describes the research design, followed by the findings and discussion, and concludes with a summary.

7.1 Purpose of the Semi-structured Interviews

The main purpose of the semi-structured interviews with key informants in this chapter is to obtain insight from domain experts of music technology in education regarding the ecologies of music software development, music software business and music education. Four domain experts were selected were music technology scholars and researchers who possessed insights into the three ecologies and could provide both insider's and outsider's views concerning linkages between ecologies. They were also in charge of teaching the music technology component in their university music and music education programmes which trained both musicians and music teachers. The focus questions developed to guide this study were:

- What music software is used in music programmes at their universities?
- What are the main problems between the design of music software and its applications in music education?
- What do you see the future of technology in music education?

7.2 Research Design

This study involved semi-structured interviews with four domain experts of music technology in education from the UK, US and Finland.

7.2.1 Participants

Four international experts of music technology in education were purposively selected and invited to participate in the semi-structured interviews. The participants were Dr Andrew King (UK), Dr Evangelos Himonides (UK), Dr Fred J. Rees (US) and Mr Matti Ruippo (Finland).

Andrew King is the programme director for the Master's in Education in Music, Technology and Education and a University Teaching Fellow at the University of Hull. He was one of the original designers of the innovative Creative Music Technology programme, which develops students' talents as performers, songwriters, composers, producers or sound engineers. He has taught various music technology related courses at the university,

including Studio Production, Advanced Production, Psychoacoustics, Studio Design, Music and ICT. He has interests in the areas of music pedagogy/psychology that relate to how learners use technology in education, and is the editor of the *Journal of Music, Technology & Education.*

Himonides is a lecturer in information technology, music and music technology education at the Institute of Education, University of London. He teaches postgraduate courses in music education, music technology and information technology at the Institute of Education and also leads the post-graduate course music technology in education.

As a musician, technologist and educator, Himonides specialises in experimental research in the fields of psychoacoustics, music perception, music cognition, information technology, human-computer interaction, special needs, the singing voice and singing development.

Rees is Professor of Music, Chair of the Department of Music and Arts Technology at Indiana University – Purdue University Indianapolis (IUPUI). He developed the first graduate music education degree distance learning programme in the United States broadcasted over the state's interactive television network. He also designed the Bachelor of Science in Music Technology degree programme that integrates music technology throughout the curriculum. His research interests include distance learning,

string education, double bass and piano performance, and music technology.

Ruippo is a Senior Lecturer in Music Technology at the School of Art, Music and Media at the Tampere University of Applied Sciences, Finland. He has led degree programmes in music and music technology since 2004, mainly teaching music education technology and Web-based music courses. His research explores the intersection between music education and new technologies, having published papers on the transformative role of technology in music education and learning, distance education technologies, synchronous learning environments and global music education networks. He is also an Apple Certified Trainer.

7.2.2 Administration of the Interviews

A formal invitation email was sent to each participant, asking them to participate and explaining the purpose of the study and the data collection. After receiving their agreement to participate, a suitable time and place were arranged for the interview. Each interview commenced with a brief introduction outlining the purpose of the study and permission was requested to audio record the interview. The interviews with King and Himonides were undertaken at their offices in London, UK. Rees was interviewed during the 30th International Society of Music Education (ISME) World Conference on Music Education, at the Thessaloniki Concert

Hall in Greece. Ruippo was interviewed at a hotel during his visit to Hong Kong. The duration of the interviews ranged from 30 to 45 minutes.

The interviews were transcribed from the audio recordings and double checked by the researcher. An email with the interview transcript attached was sent to each participant after the transcriptions were completed, thanking them for their participation and asking them to confirm the validity of the interview scripts. They were also asked to consent to their identities being revealed. All the participants confirmed the interview transcripts and agreed to their identity being released.

7.2.3 Data Analysis

Open coding (Charmaz, 2004) was used to analyse the data in this study. The researcher read all of the interview transcripts and compiled a set of recurring 'themes' in the reports. Themes were given titles such as 'use of music software in university programmes' and 'future applications of technology in music education', which were then used to categorise relevant statements for the data collection. To provide an example, a statement from a participant is analysed below:

> *On this particular course, I do not enforce the use and investigation of any particular platform. Everything we teach on this course has been designed so that it can be delivered using open source materials, trial-ware or shareware. (Himonides)*

Since Himonides' comment is related to the choice of software in his course, it is categorised under the theme of 'music software development practice'. Through synthesising interview findings with respect to the dynamics identified in previous studies on the three ecologies, additional information were added to those related dynamics as shown in Table 7. Details of the data triangulation are sourced from findings of this and previous studies, as well as literature review (also shown in Table 7).

7.2.4 Interview Questions

The questions were grouped into three categories with reference to the focus questions: (1) current use of music software in higher music education; (2) mismatch between music software development and its applications in music education; and (3) future use of music software in music education.

The experts on music technology in education were asked the following questions.

Music technology in university music programmes

- What are the core components of the music programmes in your university?
- How is music technology taught and integrated in the curriculum?
- What music software is provided for students' use?
- What software do you teach students to use?
- What factors influence the choice of software being used?

- Besides the techniques for using software, are students required to think conceptual about the use of music software/technology?
- Are there observations about students' usage of music software not covered in the questions above?

Music software development and applications in music education

- What are the limitations of today's music software when applied in classroom music teaching?
- What happens in music software development that leads to such limitations?

Future use of music software in music education

- What is the potential future or the new possibilities for music software in music education?
- What can be done in the music software development process that could help to produce better music software for educational use?

7.3 Findings

Data from the transcriptions of the semi-structured interviews were categorised into five themes:

1. Music technology courses in university programmes
2. Use of music software in university music programmes
3. Students' attitude and contextual use of music software
4. Issues of music technology applications in music education

5. Future applications of music technology in education

Theme 1. Music technology courses in university programmes

The most technology-related music programmes are the Bachelor and Master of Arts in Creative Music Technology at the University of Hull, Master of Arts in Music Education at the Institute of Education, University of London and Master and Bachelor of Science in Music Technology at Indiana University – Purdue University Indianapolis. Students are mainly undergraduate students with some postgraduate students aged between 18 and 25.

In the Bachelor and Master of Arts in Creative Music Technology at the University of Hull, there are four main strands in the curriculum: performance, composition, musicology and music technology. Students in these programmes take part in various technology-rich activities ranging from live sound reinforcement to digital instrument design. Students design their own instruments using MAX/MSP and Objective C, which covers both digital and analogue techniques. They are also involved in studio productions activities such as surround sound recording and field recording. King pointed out that many similar programmes have been launched in the UK over the past decade, but they tended to fall into one of two categories – music departments offering music technology or music and technology programmes, or engineering departments focusing on scientific aspects such as digital signal processing. The Creative Music Technology programmes at

Hull attempt to break down the barriers caused by the institutional structure and look at what the students really need.

The Institute of Education at the University of London only offers postgraduate programmes, including the Master of Arts in Music Education. According to Himonides, the programme is the only music education master's programme in the UK. A Music Technology in Education (distance learning) course was created in 2006, focusing on music technology from an educational perspective – the interrelationships between music, technology and education. Rather than teaching the skills of using technology or applying technology in education, this course enables students to be critical about using technology in an educational setting, and to evaluate the effectiveness of particular technologies (versus the use of software per se) within the classroom or educational context. The course is also available via distance learning mode.

Rees pointed out that in traditional US undergraduate music programmes, there is very little connection between individual courses and the lecturers' responsibility is to satisfy individual course requirements. The Bachelor of Science in Music Technology at Indiana University - Purdue University Indianapolis aims to synthesise separate courses to create a musicianship development programme. Every day for the first two years, three faculty members work with groups of students for an hour and a half to develop their musicianship in the music studios. Students finish the degree with what

they call the Capstone, a large project that synthesises all their learning experiences with the various music technology courses.

There are seven music degree programmes at the Tampere University of Applied Sciences, Finland. The most relevant is the Degree Programme in Music, Pedagogue in Music Technology, which focuses on the pedagogical aspects of music technology. Instead of the traditional approach to music teacher training which aims at producing regular primary and secondary school music teachers, this programme has been designed for training teachers who teach in music schools within the vocational schooling system. During the five years of study, general music courses such as instrumental skills, music history and music theory form the foundational courses in the early years of study. Advanced music technology courses and pedagogical courses are provided throughout the programme.

Theme 2. Use of music software in university music programmes

The Bachelor and Master of Arts in Creative Music Technology programmes at the University of Hull are run at the Scarborough Campus. To separate the 'noise' of the music from the main campus, the music studios are located in a building opposite the street to the main campus. The major facilities used by students are the recording studios. There are two types of recording studios – digital studios and analogue studios, with all the standard racks, microphones and other equipment. According to King, students prefer the digital studios to the analogue studios. The programme

has worked with many types of software in the past, but in recent years has narrowed the selection to specific software such as *Pro Tools, Sibelius* and *Cubase*, and specialist programmes such as MAX/MSP and CSound.

The choice of hardware and software to be taught is partially student driven. Some of the software such as *Pro Tools* were requested by students, and also reflected what was taking place in the music industry. Although the decision was partially market-driven, the programme aims to expose students to a broad range of software, helping them to understand the limitations of the software, and enabling them to adapt their knowledge to any studio environment, effectively navigating around a variety of systems.

> *It is partially student-driven. Some of the software platforms such as Pro Tools are what the students are asking for. That reflects what's going on in the music industry. We try to teach the principles through the software, but also consider what our students want and are aware of as well. It is partially market-driven but what we try to give them is a broad range of software platforms so that they can understand the principles of the software and be adaptive to any studio environment, navigating around those systems. (King)*

The music technology course at the Institute of Education, University of London, is focused more on the philosophical aspects of using technology in general than the practical usage of particular software; thus it does not require the use and investigation of any particular software. According to Himonides, the music technology course addresses 'the need to separate heuristics and learning how to use particular tools or software, and what can be done with different kinds of tools and software'. The course emphasises

the use of open source materials, trial-ware or shareware in music education such as *Ripper* (freeware) for teaching sequencing, digital analysis tools or real-time digital signal processing. It also help students to know that music can be represented by several system such as staff notation, midi protocols and waveforms, as well as understand the characteristics of music by visualising the sounds and design through graphic representations. Thus, contextualising or conceptualising music technology and software is focused in the Master of Arts in Music Education.

> *On this particular course, I do not enforce the use and investigation of any particular platform. Everything we teach in this course has been designed so that it can be delivered using open source materials, trial-ware or shareware. (Himonides)*

In the Bachelor of Science in Music Technology at Indiana University - Purdue University Indianapolis, faculty members work with the students in the music studios for two years to develop their musicianship. The software currently used during the development period or beyond include *Sonar*, *Logic*, *Pro Tools*, *Reason*, *Sibelius* and *Ableton* Live.

The choice of hardware and software is both people-driven (by faculty members and students) and market-driven. For example, in the past students were taught to use Sibelius as the choice software for notation. Then the head of MakeMusic – the company that produce *Finale* – joined their board of advisors, so they switched to *Finale*. They had been teaching *Sonar* as the sequencing software because the course instructor preferred *Sonar*, but they

changed it because *Logic* and *Pro Tools* are the industry standards. The principle is that the students should become familiar with the software before going out and working with it. Thus, the decision is generally made according to what is current in the industry. There are some differences between software of the same kind – for example, *Sonar*, *Pro Tools* and *Logic* – but the difference are not that great. However, students may find difficulties in transferring their knowledge between software, so the preference is to teach software that is widely used in the industry.

> *The decision is generally made by what's going on in the industry. There are some differences between Sonar, Pro Tools and Logic, but not a lot. But the reality was that the students only used Sonar for 4 years, and then they went out and figured out that they had to use Pro Tools, and they hadn't seen it before. The transfer of knowledge would be a problem to them. (Rees)*

A range of software is studied throughout the Degree Programme in Music, Pedagogue in Music Technology at the Tampere University of Applied Sciences, Finland, including *Pro Tools*, *Logic*, *Reason*, *Ableton Live* and *Sibelius*. As the programme focuses on the pedagogical aspects of music technology, it also includes the study of pedagogical music software such as *GarageBand*, *iMovie* and online resources. The choice of particular software for teaching is based on the practical application of the software in the industry. For example, *Sibelius* was chosen as the music notation software because it is the leading brand for notation software, while *Pro Tools* is the industry standard for digital audio workshop. In addition, the

School of Art, Music and Media is a certified training centre for Apple *Logic*, hence *Logic* is also included in the curriculum.

Theme 3. Students' attitudes and contextual use of music software

King mentioned issues regarding students' usage of both analogue and digital equipment and recording techniques. Students who use analogue equipment are expected to learn the principles of signal flow, routing and monitoring and apply these to their listening skills when engaged in recording situations. Hence they should be relying on their musical ears than the data shown on the computer screen. On the contrary, students who use digital equipment tend to rely on information shown on the screen than on their ears. Students have shown preference towards the use of digital equipment as digital processing is supported by an auto-correcting mechanism that fixes errors, hence is more 'forgiving' than analogue equipment in certain aspects such as the use of reverbs in insert channels. While errors can be heard when done in the analogue mixing studio and the students have to fix these themselves, the automated functions in digital mixing studio software would fix these by default.

> *One thing that I found was the misunderstanding in the digital studio because it can be a lot more 'forgiving' in aspects like using reverbs in inserts, which are rarely done in the analogue world. If you do it in the analogue mixing studio you can hear it is incorrect but the software will fix it for you in the digital mixing studio. (King)*

The same phenomenon occurs in students' use of music software. For example, music notation software such as *Sibelius* will point out the technical errors when the user writes out of the pitch range. Students then get the wrong impression that everything not pointed out as an error by the software is acceptable. If uncorrected by the software, students may write a complex clarinet melody that is unplayable by a clarinet player in reality, but the melody can be performed in its entirety with playback using the software synthesiser.

> *Same as Sibelius. It will highlight in red for you if you are writing out of the range. People now write complex clarinet parts that cross the bridge, which is unplayable to a clarinet player, but Sibelius could play it for you. (King)*

Himonides pointed out that practitioners in the field of music education have reservations about the use of technology in education. Many music teachers perceive themselves as traditional or non-technical, and hold the attitudes of digital immigrants.

> *We are as much technological by default as we are musical by default. You will have many people that perceive themselves as or have the attitudes of a digital immigrant. (Himonides)*

Himonides also pointed out that many teachers are very concerned about how to use a particular technology or software, but they do not understand why they need to use it, or how things are going to change, what will be the value-added, how music education could be better with it, or how it could be

used to facilitate better musicians. They focus on the actual tools rather than the reasons behind their usage.

> *Many people are concerned about how to use a particular tool and they do not have an understanding about why they need to use that tool, or how things are going to change, what is the value-added, how are you going to make music education better, or facilitate better musicians. They focus on the actual tool. (Himonides)*

Theme 4. Issues of music technology applications in music education

All the participants agreed that there are limitations in current music software. King pointed out that every music software developer or company has its own strength and weakness, and teachers or students should not rely on any single software. For example, *Cubase* is good for MIDI sequencing but not for producing notation of a publishable standard, while *Logic* is good for audio recording and editing but not suitable for MIDI sequencing. Ruippo pointed out the language and cultural problems resulting from the dominant Anglo-western culture in music software development. For example, Finnish language is not available in most of the music software, and Chinese instrumental sounds are not included in most sound libraries of digital audio workshops software.

> *Software developers such as Sibelius are very good at attending music education conferences, talking to music teachers to see what they want. Cubase, on the other hand, also has plug-ins for teachers, but is aimed very much at pre-university level and doesn't always map onto what's going on in the national music curriculum. (King)*

The ease of use in software was another issue highlighted by the participants. King pointed out that the difference in the perspectives of software engineers and music teachers leads to the difficulties that teachers experience in using the music software. Some software packages seem to be over-complicated as the gold-plating practice (whereby software developers add unnecessary, frivolous functions or requirements to the software) means having many functions that are not used by music teachers or even music industry practitioners. Himonides agreed, explaining that software developers do not think like their targeted users and music software development cannot be led by software developers because they may not have a clue about real-world practice. Rees brought out that some technological platforms such as iPad, have restrictions on what can be developed within the platform. For example, iPad does not allow the attachment of external devices such as a foot pedal or MIDI keyboard. More useful music software or functions such as a page turner or pedalled piano could be developed if there were no such restrictions.

> *Software developers don't think like their targeted users. They will not make mistakes when using software. This is one of the problems in software development. Because the thing will become something that was perceived by a group of people that are completely outside the context that they want to target the software to. (Himonides)*

Theme 5. Future applications of music technology in education

King pointed out that the future development of music technology in education will be towards design, whereby musicians apply their own ideas

to the design of their own instruments. This view is echoed by Himonides, who supported the involvement of music teachers in designing music software for education. Ruippo called attention to the difference in attitudes towards music software between software developers and music teachers, in which teachers' pedagogical thinking may not be shared by developers. While music teachers may not necessarily use every part of any music software, Ruippo suggested that the future direction of music software development for educational purposes would be the extension of single-task mobile apps, which serve only one purpose but have the advantage of convenience.

> *I think the way it is going is towards the design. To me the real cutting-edge in terms of the software seems to be the musicians bringing their own ideas to design their own instruments – more user driven in a sense. (King)*

> *The discourse between educator, user, software designer, software developer and the music industry needs to become much more open. (Himonides)*

King forecasted that the use of software that are able to assess or give useful feedback on students' music performance (such as pitch and rhythmic accuracy) would become more widespread. Existing software that can provide basic level of feedback on musical performance would continue to develop into 'intelligent tuition systems' that can also provide meaningful feedback to help users improve their instrumental or vocal skills. But Himonides identified the lack of communication between the end users and

software developers to be a huge obstacle to the further advancement of such performance assessment technology. Communication between participants belonging to different disciplines within the ecology of software development, music education and business practice must be improved. He argued that the multi-dimensional issue to be addressed is the discourse and the unclear balance and relationships between educators, users, software designers, software developers and the music industry. Music software design still relies heavily on the expertise software engineers, and music teachers have to fit into the pre-determined design of particular software which does not consider their particular needs.

> *The difficulty with computers is giving meaningful feedback. We can always get data from the software, but how to interpret the data is another thing. (King)*

> *There is no issue about copyright, software protection. It is all about communication, both for music educators and computer scientists. (Himonides)*

> *They [music educators and software developers] need to have equal voices and participate in the discussion. Otherwise you would come up with either inappropriate or non-effective or non-meaningful designs. (Himonides)*

Both Himonides and Rees proposed that a universal standard is needed to improve music software development. They gave XML and MIDI protocols as examples of how different software can communicate with each other without sharing deliverables. Rees described the proposed universal standard as a value system – at a lower level it is about getting the right pitch

or writing the right harmony, and at a higher level it is about which timbre or harmony progression sounds 'good', or what expressions should be used in particular situations. Thus, the design of music software should be built on some appropriate value system.

> *I see the future of software being completely open platform, completely extendable. You can use an XML base where any software can speak to any other software if they do not have to share deliverables, exchange meaningful information about what the product is all about. (Himonides)*

> *Just like the idea of the MIDI protocol. You create a piece of hardware or software that can do certain things – detecting the pitches, rhythm, timbre – and it will just do that regardless of what programme you are working with. What we call a universal protocol. (Rees)*

7.4 Discussion

7.4.1 Music Technology in University Music Programmes

The criteria used by domain experts in selecting music software for teaching experts were three-fold: student-driven, teacher-driven, and market-driven. Student-driven and teacher-driven factors echoed findings in the previous chapter, in which the same criteria were found in music teachers' choice of music software to be applied in their teaching. To prepare music or music education students for real-world practices, the considerations on what is going on in the music industry with respect to the use of music technology have been included in university music programmes. Music education

programmes therefore have been providing knowledge of up-to-date music technology to music teachers in training. Due to time and resource constraints that could only prepare music education graduates at a basic level, teachers need to undertake on-going professional development to keep abreast of the latest technology. The provision of software training beyond graduation is an important role that music software retailers could play.

All participants agreed that students have overly focused on the technology itself, neglecting the more important and fundamental reasoning of why and how they use the software. Even though they were young, many tended to label themselves as digital immigrants and some even tried to avoid using technology. They aimed at getting the assignments done rather than achieving what they should learn – for example, producing a loop-based music using music sequencer by trial and error, or producing a music score using music notation software that may sound acceptable in synthesiser simulation without considering how musicians will play it. In fact, today's technologies can give students the false impression that their lack of knowledge is unimportant and hence they may not be serious about course content involving music theory, basic recording principles, and orchestration techniques. Music software nowadays emphasises the power of production and enjoyment of music within a short time, offering impressive results with simple clicks and without the need for in-depth knowledge about music. More importantly, students tend to ignore the

fundamental reasons regarding why they make music, or what they have learnt through the music making process. This phenomenon concerning the use of music software has been raised by Gouzouasis (2005) who called attention to the use of music technology without formal guidance and supervision that would result in the distortion of the authenticity of music education. Upitis (2001) also highlighted the blind acceptance of music technology in the belief that technology can enhance students' learning without examining the evidence of the value that it can bring to students.

Music software could be useful tools for facilitating music teaching and learning, but those who use these without removing their misconceptions about music software in education are likely to end up with adverse results. To avoid this, music teacher education should not only focus on the technical skills of using music software, but the underlying principles of why and how software should be used in music classrooms. Interview data have shown at the master's level the focus of music technology courses is more about theory and philosophy than practice, and the reverse at the bachelor's level. While a balance between the two is desirable, the constraints of curriculum time may not allow for it in reality. Hence ongoing professional development is vital for practicing music teachers to maintain a sustainable career.

7.4.2 Limitations of Music Technology

The participants commonly agreed that all music software has particular limitations (see Section 6.5.6). The restriction of having only the English language in commercially available software has been pointed out by a domain expert from Finland. This issue has not been brought up earlier since the participants came from countries where English is the official or co-official language (as in Hong Kong). Glocalisation strategy is one of the successful factors adopted by successful global business, which is also applicable to software business (Svensson, 2001). In the last chapter, views from music software retailers have brought out the importance of glocalisation practice in music software business, but noted that the music software industry are more concerned about globalisation than localisation. This is also echoed in the study of the ecology of music education, known as the 'Western canon' phenomenon (see Section 6.5.4). Glocalisation offers one important strategy to seek new users group while at the same time meeting the needs of local markets. This not only provides new business opportunities but also enhance the learning opportunities of locals with music technologies.

All participants held similar views about the limitations and weaknesses of different music software, and particularly about not relying on a single software. They explained that software developers do not have the necessary knowledge of music education practices and are uninformed about the needs of music teachers when designing music software. They emphasise the need for open communication between music software

developers and end users, i.e., music teachers, and to strike a balance between the voices between the two stakeholders in the software development process. Bringing music teachers' involvement in the software development process adds interdisciplinary value and advantage to the development team. Similar views were shared by music software developers in Chapter 4, who addressed the importance of users' feedback from music teachers, and connected with the role of music software retailers as a communication channel between developers and music teachers in Chapter 5. This echoed with Leong's (2002) observation that music software development should be interdisciplinary, bringing expertise from different fields together.

7.4.3 Future Applications of Music Technology in Education

The transformative potential of technology in music education is recognised by the four domain experts and music software developers in the ecology of software development (see Section 4.5.4) but not by music teachers in the ecology of music education (see Section 6.5.4). The of literature review has indicated that the role of music technology has changed from being an assistive tool (Savage, 2010; Southcott & Crawford, 2011) that reinforces the traditional music curriculum to become a transformative tool (Wise, 2010; Wise et al., 2011; Ward, 2009). This has impacted in transforming the curriculum from being characterised by the traditional instructivist (teacher-centered) approach to having a constructivist (student-centered)

approach in music education. In-depth investigation into how music software is designed is needed to make sure that the right tool is being produced for facilitating desired changes in music education.

With regards to future applications of music technology in education, participants envisage musicians and music teachers bringing their own ideas to the design of music software. This view further extends the involvement of music teachers in not only providing domain knowledge, users feedback and experience (as shown in Section 4.5.5), but also in determining the purpose of the software – achieving an interdisciplinary software development approach in music software design. The importance of having an interdisciplinary music software development team has been recurring theme in the literature and findings reported in previous chapters. If the interdisciplinary team cannot be formed, individual ecologies would continue to retain their key operational practices and their contributions cannot be synergised to achieve a greater combined effect.

7.5 Summary

The semi-structured interviews with domain experts of music technology in education elicited their conceptual and philosophical perspectives. Their interview data have provided an overarching view of the three ecologies, which confirms data from the studies of the three ecologies. The literature

review also provides confirmation of the two sets of data, resulting in data triangulation as shown in Figure 17.

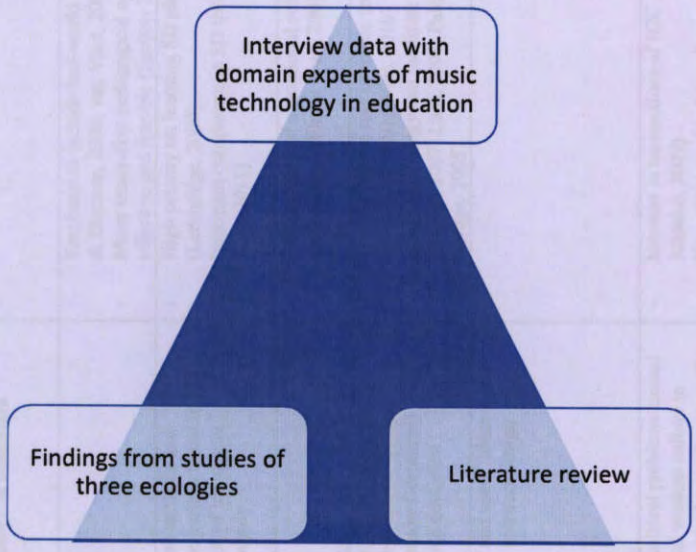

Figure 17. Data triangulation of findings with studies of three ecologies, interview data with domain experts and literature review

Findings from the studies of three ecologies, interview data with domain experts and literature review triangulate with each other to enhance the validity of the conclusions in this thesis. Table 9 shows the summary of the data triangulation in this thesis.

Table 9

Data triangulation with studies of three ecologies, interview data with domain experts and literature review

	Studies of the three ecologies: Software development (SD), Software business (SB), Music education (ME)	**Interview data with domain experts of music technology in education**	**Literature review**
Curriculum Development	Ecology of SD: - Curriculum updated frequently - Innovative pedagogical approaches not found in SD training		- Emphasis to include real-world practice in SD training (Huang & Distante, 2006; van Vliet, 2006) - Many innovative pedagogical approaches being shown to be effective and feasible (Section 2.1.1)
Software development training	Ecology of SD: - Learning SD considered to be important by software engineering lecturers - Only introductory level of software development taught in SD training	- Poor requirement analysis and gold-plating phenomena found in SD reflected the lack of training in requirement analysis	- High priority on learning SD placed by software professionals (Lethbridge, 2000) - Important component in SD training being neglected (Bareiss & Katz, 2011)
Adherence	Ecology of SD: - Music software developers have their own ways to compensate for lack of formal SD training		- Development of educational software rarely follows any SD methodology (Flores et al., 2001)
Insight and foresight	Ecology of SD: - Music software developers' insight into end users' needs; foresight in addressing the impact of informal music learning	- Transformation power of technology from instructivist (teacher-centered) to constructivist (student-centered) approach in music education	- Transformative role of music technology in education (Beckstead, 2001; Wise, 2010) - Music technology can facilitate informal music learning (Finney, 2007; Lum, 2008; Palmer & de Quadros, 2012; Savage, 2005)
Value-added involvement	Ecology of SD: - Music software developers' recognition of music teachers' needs; working closely with them to develop music software - Provision of users feedback, experience, domain knowledge and pedagogical needs to music SD process by music teachers	- Involvement of end users' ideas in the future of music software design	
Software business practices: - Product delivery - Marketing - Glocalisation	Ecology of SD: - Demo versions of music software to attract new users - Online and traditional marketing strategies by music software developers - Music software developers selling software directly to end users via the Internet - Music software developers' foresight to attract new user groups helps surviving in the SB market Ecology of SB: - Increase in download (virtual) and decrease in boxed (physical) software sales - Marketing of music software more focused on professional users' needs - Music software retailers face the same challenges and	- Language and cultural problems caused by the dominated western culture in music software development – currently available music software is globalised, but not localised	- Increase in transactions of B2C model in SB (Valtakosi & Rönkkö, 2009) - New users want to know how software can solve their problems (Lenberg, 2010) - Glocalisaion strategies adopted by successful global companies (Han, 2008; Koller, 2007)

	opportunities globally		
Version Control	Ecology of SB: - Rapid software updates and releases of new versions		
Users' Feedback	Ecology of SD: - provision of users feedback, to music SD process by music teachers Ecology of SB: - Music software retailers serve as the main communication channel between developers and end users	- Lack of communication between music software developers and end users	
Software training and users support	Ecology of SD: - Provision user support to end users by music software developers Ecology of SB: - Music software retailers' measures to provide software training and user support	- Music students classified themselves as non-technical and digital immigrants; reluctant to seek for support	- Music software beyond technical expertise of schools technology expert (Noxon, 2003) - Music software too complex for non-expert users (Lenberg, 2010)
Competence for teaching with teaching	Ecology of ME: - Provision of content knowledge and technological competence to music teachers by music teacher training	- Music teachers' lack of fundamental reasoning on why and how to use music software - Insufficient pedagogical skills of music teachers to incorporate music software in teaching	- Provision of pedagogical skills, content knowledge and technological competence and cultivate music teachers' positive attitude towards music technology by music teacher training (Section 2.7.2)
Choice of music software	Ecology of ME: - Choice of music software for use based on school, human and software factors	- Choice of music software for use are teacher-, student- and market-driven	
Facilities and support	Ecology of SB: - Music software retailers' measures to provide software training and user support Ecology of ME: - Availability of facilities and support affects teachers' use of music software		- Limited software, computers equipment and related resources inhibit teachers to apply music technology in teaching (Busen-Smith, 1999; Roblyer, 2006)
Software design and music teaching	Ecology of SD: - Requirement analysis not seriously being addressed in music software development process as reflected by software engineering lecturers Ecology of ME: - Effect of comprehensiveness of software functions and quality of software output to the choice of music teachers to use music software - Music teachers' desired software functions not addressed by software developers - Narrowed array of sonic landscape in current music software	- Limitations and weaknesses exist in different music software hence cannot rely only on one single software	- Effect of quality of software on ones' willingness to use music software (Airy & Parr, 2001) - Importance of software engineering education to the quality of software (van Vliet, 2007) - Lack of considerations in requirement analysis resulted in poor interactions between music software and users (Cooper et al., 2007), and music software viewed as technically complicated by non-expert users (Lenberg, 2010) - Insufficient interdisciplinary teamwork and user-oriented considerations between music software developers and music educators (Flores et al., 2001; Krüger et. al, 1999; Leong, 2002) - Music software was built based on western traditional music theory (Beckstead, 2001; Webster, 2011b)
Pricing factor	Ecology of SB: - Discounts on educational versions of music software Ecology of ME: - Effect of pricing on music teachers' choice of software		

Tables 9 shows two different levels of data triangulations presented in this study – the triangulation of findings in the studies of the three ecologies (see Section 6.6); and the triangulation with those findings, interview data with domain experts and literature review. The former level of triangulation may not be comprehensive enough to triangulate all the dynamics being identified. The latter level supplements such deficiency and strengthens the validity of the entire research project.

This chapter has presented the findings from a series of semi-structured interviews with domain experts of music technology in education. In general, their views concur with earlier findings of the three ecologies in Chapter 4, 5 and 6 and show how the ecologies of music software development, music software business and music education interconnect with each other. The next chapter will present a model of sustainable ecosystem consisting of the three ecologies.

CHAPTER 8

Model Development and Conclusion

This chapter describes the development of a sustainable ecosystem consisting of the ecologies of software development, software business and music education. The model developed in this study is a conceptual model that describes salient aspects of the physical and social world for the purposes of understanding and communication (Mylopoulos, 1992). The model can help us understand how each of the three ecologies of software development, software business and music education work in reality, the key components and dynamics of each ecology and how the three ecologies connect and interact with one another, and thus inform the possibilities for creating a sustainable ecosystem. The model is presented at three levels of abstraction, with each deeper level of abstraction representing a more detailed description.

This chapter starts with the scope and limitations of the model development, followed by descriptions of the model at the first, second and third levels of abstraction. The first level of abstraction focuses on the physical delivery between ecologies, the second focuses on the interdisciplinary knowledge transfer between the three ecologies and the third addresses key considerations for a sustainable ecosystem, including threats to

sustainability within and between the three ecologies in question, recommendations for the way forward.

8.1 Scope and Limitations

As a model is an abstraction of reality, the process of model development revolves around the abstraction of realities. The abstraction defines the scope of the model and requires the identification and extraction of the key elements from real-world dynamics within each of the three ecologies of software development, software business and music education.

The scope is necessarily bound by the primary activities within and between the ecologies as described in chapters 4 to 7. The model shows how the key dynamics within and between each ecology affect individual ecologies and the overall ecosystem. The model provides insight into how certain dynamics should be addressed so that the entire chain of activities governing software development and software business can better contribute to the integration and appropriate application of technology in school music education.

It is acknowledged that any study incorporating technology involves rapid and dynamic changes, and this study is limited to the technology being produced and used during the time it was undertaken.

8.2 First Level of Abstraction

Interview data from previous chapters revealed how music software designed by music software developers are distributed by the software retailers and purchased for use by music teachers. As the first step in understanding how the ecosystem works, the model at the first level of abstraction is presented from the perspective of the physical delivery between the ecologies.

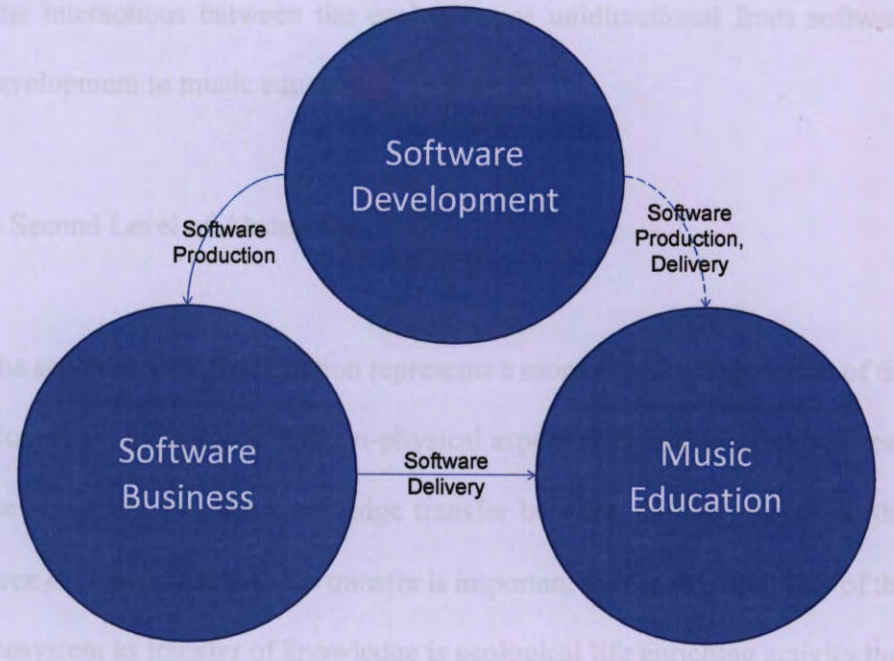

Figure 18. Ecosystem at the first level of abstraction

The first level of abstraction (see Figure 18) simplifies the interactions between the three ecologies by including only physical contacts, which treats the ecologies themselves as individual entities. Software developers

produce music software and deliver it through a software company to the music education sector. Through the emerging B2C market, software developers are able to deliver their software product directly to the music education sector.

At this level of abstraction, the ecology of software development acts as the producer of music software, and the ecology of the software business is the 'middleman' who distributes the music software to the music education sector. Music software is 'consumed' in the ecology of music education. The interactions between the ecologies are unidirectional from software development to music education.

8.3 Second Level of Abstraction

The second level of abstraction represents a more detailed description of the ecosystem, focusing on the non-physical aspect of the interactions between the ecologies, i.e., the knowledge transfer between the key players in the three ecologies. Knowledge transfer is important for the sustainability of the ecosystem as transfer of knowledge is ecological life enriching activity that connects the key players and the three ecologies together (as shown in Figure 19).

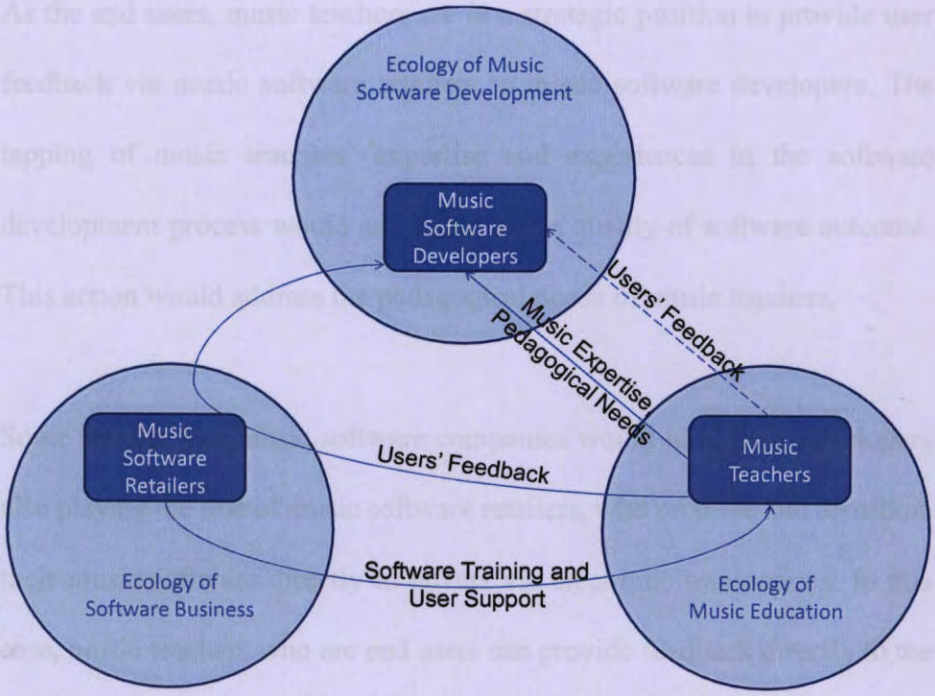

Figure 19. Ecosystem at the second level of abstraction

The second level of abstraction (see Figure 19) focuses on the knowledge transfer between key players in the ecologies of software development, software business and music education. The key players are music software developers, music software retailers and music teachers. Music software retailers act as the middleman in the physical delivery of music software, playing a key role as the main communication channel between music software developers and music teachers. Music software retailers also provide software training and user support to music teachers, and continue to professional support to music teachers with the most up-do-date music technology competence and knowledge.

As the end users, music teachers are in a strategic position to provide user feedback via music software retailers to music software developers. The tapping of music teachers' expertise and experiences in the software development process would add value to the quality of software outcome. This action would address the pedagogical needs of music teachers.

Some smaller size music software companies would have their developers also playing the role of music software retailers, who promote and distribute their music software directly to buyers via electronic transactions. In this case, music teachers who are end users can provide feedback directly to the music software developers via electronic means such as email or the companies' official websites.

8.4 Third Level of Abstraction

At the third level of abstraction, the findings from the studies of the three ecologies (Chapters 4 to 6) as well as the perspectives from domain experts of music technology in education (Chapter 7) informed the construction of a detailed model of the ecosystem (see Figure 20). The ecosystem consists of the key players, activities and dynamics in the ecologies of software development, the software business and music education.

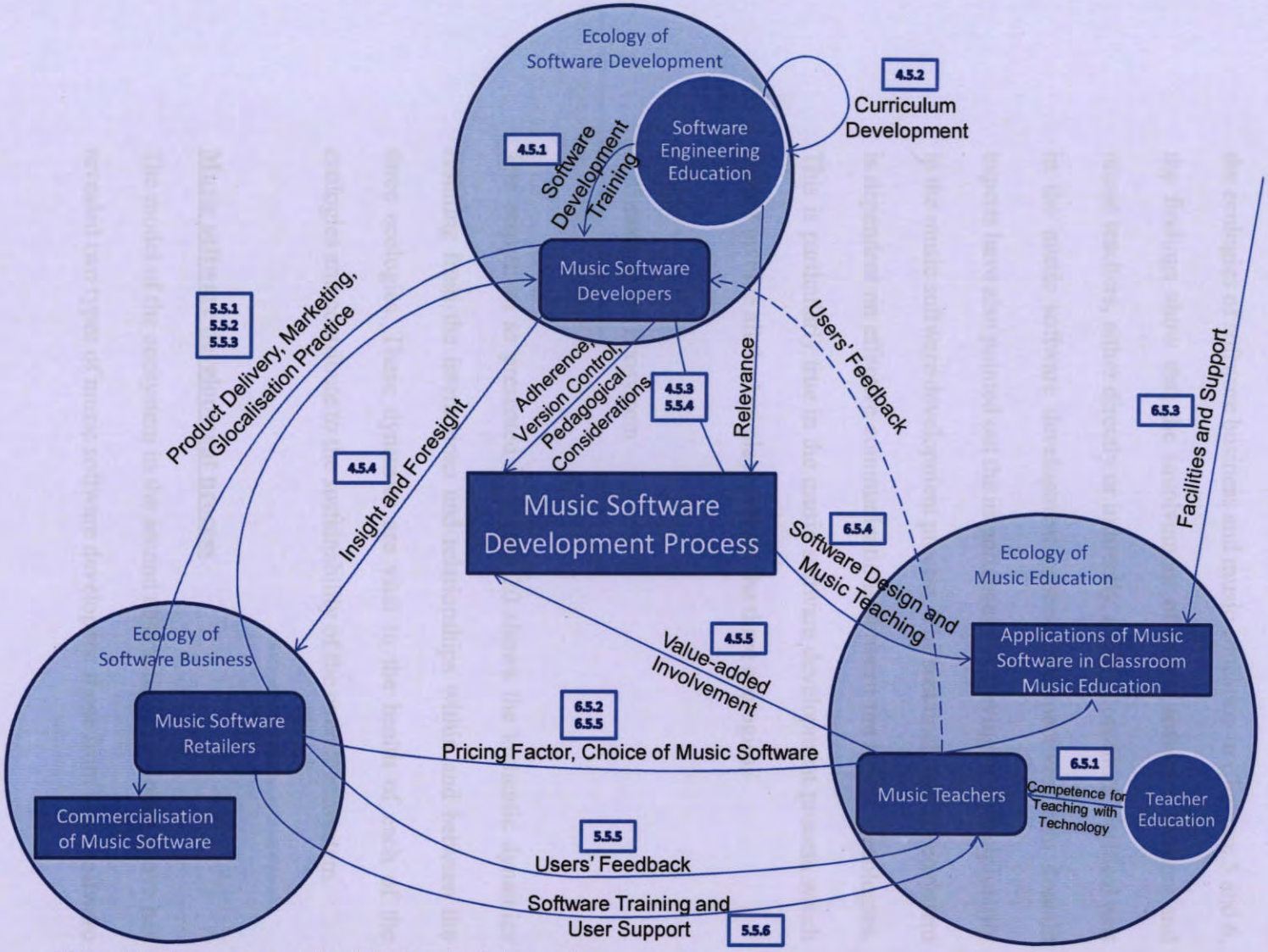

Figure 20. Ecosystem at the third level of abstraction

In the study of the ecology of software development in Chapter 4, it was uncertain whether the music software development process should be treated as an activity within the ecology of software development or as an interdisciplinary activity at the centre of the ecosystem. After the studies of the ecologies of software business and music education in Chapters 5 and 6, the findings show that the involvement of music software retailers and music teachers, either directly or indirectly, should constitute a critical part in the music software development process. Perspectives from domain experts have also pointed out the importance of achieving interdisciplinarity in the music software development process. The sustainability of ecosystem is dependent on effective communications between the inherent ecologies. This is particularly true in the music software development process which should involve all the key players from the three ecologies.

8.5 Threats to the Ecosystem

The ecosystem as presented in Figure 20 shows the authentic dynamics resulting from the interactions and relationships within and between the three ecologies. These dynamics are vital to the health of each of the ecologies and contribute to the sustainability of the overall ecosystem.

Music software development process

The model of the ecosystem in the second abstraction mentioned above has revealed two types of music software developers: those from large software

companies that rely on software retailers to promote and distribute their software products and those from small software companies who also play the role of promoters and distributors for their software products. The first type of music software developers has sufficient expertise to develop large-scale music software; however their targeted customers are professionals who work in the music industry. Without getting input from music teachers outside the music industry, these developers are not likely to have the necessary insight and foresight to cater for the growing educational market for music software. As their development work have little to do with music teachers, the music software development process would not have given adequate considerations to the specific needs of music teachers in school contexts. This gap between music software developers and music teachers could be bridged by music software retailers. Because the retailers have not taken up this potential role, the disconnection between developers and teachers has not been resolved.

The second type of music software developers works more closely with music teachers in developing music software. However, their software development teams are smaller in size and they may lack of formal software engineering training, resulting in the formal software development process not being implemented.

Music software business

The main roles of music software retailers in a sustainable ecosystem are to distribute music software products from music software developers to music teachers and act as a communication channel between the two. To achieve these, they need to establish close links with music software developers for promoting and distributing their products as well as with music teachers who are end users of music software produced by developers. Without the proactivity of retailers, music teachers who would not integrate technology in school music education may remain in their isolation from the knowledge base available from retailers. With the availability of online purchases, retailers may not be in direct contact with teachers who use software in their teaching, and lose opportunities to gaining intimate user feedback regarding how software are used in particular education contexts. Such insider information is valuable for both software retailers and software developers.

The commercialisation of music software requires such information which comes through knowledge transfer between software retailers, music teachers and software developers. A successful music software company require insider's insight to gain business foresight into how music software should be developed and refined to cater for end users' needs as well as inform their business strategies and strengthening their competitive advantage.

Applications of music software in school education

Music teachers possess the basic competence to choose and apply music software in their teaching because of the training received through music teacher education. There are internal threats inside the ecology such as unavailability of facilities and technical supports from schools, and music teachers' insufficient technical competence and pedagogical skills to apply music software in their teaching. The training of technical competence provided by music teacher education and on-going professional development is insufficient to prepare and update music teachers to use and apply music software in their teaching.

Another threat to the ecology of music education is the lack of value-added connections between music teachers, music software developers and music software retailers. While music teacher education is not able to provide comprehensive training of technological competence because of time constraints and tight programme content, music teachers have to update themselves with the latest music technology knowledge and skills. However, the training and user support provided by music software retailers seem not to be fully utilized by music teachers for a kind of professional development, resulting in the knowledge gap between music teachers and the latest practices of music software in education.

8.6 Sustainable Ecosystem: The Way Forward

Recommendations for creating a sustainable ecosystem that embraces the ecologies of software development, software business and music education are made below.

Music software developers

Music software development process is the key activity that determines the sustainability of the ecosystem. It is an interdisciplinary activity that involves all key players in the three ecologies. Music software developers being the leading position in the development process should ensure the formation interdisciplinary software development team. Large music software companies should address the needs from music teachers; their requirement analysis process should not only elicit functional requirements from music teachers, but also understand the music teaching and learning process in the classroom context.

Smaller music software companies who work more closely with music teachers should ensure the adherence of their music software development process, i.e. the best practices of all the parts in the software development process. This is to ensure the software product is authentic, high quality, and relevant to its applications in music education.

Music software developers should be interested in both the music teachers' curriculum and pedagogical needs when designing their software for music education purposes. They should ensure that the software being produced

could support teachers' music teaching process as well as curriculum requirements.

Music software retailers

To enhance the connections between music software developers and music teachers, music software retailers should reach out to cater for music teachers' needs in their software retailing business practice. This includes the promotion of music software focusing on how the software can solve music teachers' problems and how the software can facilitate their teaching, as well as provide software training workshops for music teachers to update their music technology knowledge.

In relation to music software developers, music software retailers should provide knowledge about the considerations of software business in the software development process. Strategies such as version control, glocalisation could help software developers to better sell their products, as well as benefit the end users such as provision of software in language other than English.

Music teachers

Music teachers should keep abreast of new developments in educational research and the software industry by establishing connections with music software retailers and music software developers. They can benefit from updating their technological competence of using music software from

professional training provided by music software retailers as well as acquiring the most up-to-date knowledge about music technology.

Music teachers should be willing to provide education-specific feedback on music teaching and learning to music software developers for them to improve the design of music software. This would increase the relevance of music software in the school music education context. The connections between music software developers, music software retailers and music teachers could nurture the interdisciplinary teamwork of the music software development process.

Domain experts of music technology in education

Domain experts of music technology in education comprise researchers and scholars with expertise music, technology and education. They can play a significant role in the sustainability of the ecosystem by undertaking interdisciplinary research with music software developers, retailers and music teachers. These could involve examining and evaluating software development models adopted by software developers and ways to enhance the functional roles of music software retailers in the ecosystem. They could also investigate the effectiveness of teachers' usage of music software in relation to limitations in software design.

8.7 Conclusion

This thesis has investigated the three ecologies of software development, software business and music education, identified the primary dynamics within and between these ecologies. It has described how the three ecologies interconnect with one another to form an overall ecosystem as well as the key threats to ecological sustainability. The thesis has achieved its goal of developing a model of sustainable ecosystem that takes into consideration these threats and has made recommendations for the creation of a sustainable ecosystem that is mutually beneficial for each of the three residents. This study has contributed to filling the gap in the literature review as no ecological approach has been found to have examined the areas of software development, software business and music education.

This thesis has clarified the primary activities and dynamics in each of the three ecologies, and identified the main threats to the overall ecosystem that supports the design and production of music software, distributed by music software retailers, and purchased by music teachers for applications in class music education. If the threats are ignored, individual ecologies would retain their traditional operational practices without harvesting the benefits of inter-ecological knowledge transfer and depriving themselves of achieving synergies that create a greater combined effect.

By revealing the dynamics vital to the three individual ecologies and threats to the overall ecosystem, the sustainability of the ecosystem can be enhanced through strengthening the dynamics and removing the threats.

Key players in each of the ecologies should be aware of the latest development in their ecologies and update themselves with relevant skills, practices and knowledge. Between the ecologies, the success of creating a sustainable ecosystem lies in having two conditions achieved – interdisciplinarity and knowledge transfer.

Interdisciplinarity is a key condition for achieving a sustainable ecosystem. As the core activity in the ecosystem, the music software development process needs the cooperation between music software developers, music software retailers and music teachers to share knowledge pertaining to their respective ecologies and contribute to interdisciplinary software development.

Knowledge transfer between key players is necessary for ensuring effective interconnections being achieved between the three ecologies in which insider knowledge are possessed by key players within each ecology. These should be shared for harnessing the benefits of unfolding the possibilities afforded by such aggregate effort. This includes the quality of training received by software developers and the quality of teacher training received by music teachers.

The model developed by this research has revealed how music software can be better designed and delivered more effectively via music software retailers for more widespread applications in school music education. The

extent to which interdisciplinarity and knowledge transfer can be realized would impact on the sustainability of the ecosystem which serves the mutual needs of the three ecologies of software development, software business and music education. It is hoped that this research would stimulate the key players in the three ecologies to be more mindful of the dynamics that promote ecological soundness and sustainability as well as to be engaged with future interdisciplinary research in this area.

8.8 Further Studies

A natural follow up to this research is in-depth investigation into the values underpinning the operational practices of key activities within each of the three ecologies. This might include examining the technical details of music software development from the perspective of software engineering, marketing research undertaken by music software businesses and their business strategies, the pedagogical approaches of music teachers in relation to integrating music software/technology in school music education, and the correlation between beliefs about the effectiveness of music software in music education and teacher factors such as educational background and teaching experience. Other studies may focus on key stakeholders' conceptions and valuing of interdisciplinarity and knowledge transfer as well their affinity towards the notion of ecological soundness and sustainability.

References

Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. L. (2004). *Guide to the software engineering body of knowledge* (SWEBOK). IEEE.

Agarwal, B. B., Tayal, S. P., & Gupta, M. (2010). *Software Engineering & Testing: An Introduction.* Hingham, MA: Infinity Science Press.

Ahmed, F., & Capretz, L. F. (2007). Managing the business of software product line: An empirical investigation of key business factors. *Information and Software Technology, 49*(2), 194-208.

Airy, S., & Parr, J. M. (2001). MIDI, music and me: Students' perspectives on composing with MIDI. *Music Education Research, 3*(1), 41-49.

American Marketing Association (2004). *Definition of Marketing.* Retrieved September 15, 2013, from http://www.marketingpower.com/

Anewalt, K., & Polack-Wahl, J. A. (2010). Teaching an iterative approach with rotating groups in an undergraduate software engineering course. *Journal of Computing Science in Colleges, 25*(6), 144-151.

Anisetty, P., & Young, P. (2011). Collaboration problems in conducting a group project in a software engineering course. *Journal of Computer Science in Colleges, 26*(5), 45-52.

Baker, M. (2000). The roles of models in artificial intelligence and education research: A prospective view. *International Journal of Artificial Intelligence in Education, 11*(2), 122-143.

Barker, M., & Inoue, K. (2009). IT SPIRAL: A case study in scalable software engineering education. In the *Proceedings of 22nd Conference on Software Engineering Education and Training* (pp. 53-60). Hyderabad, India: IEEE Computer Society.

Bareiss, C., & Katz, E. (2011). An exploration of knowledge and skills transfer from a formal software engineering curriculum to a capstone practicum project. In J. B. Thompson, E. O. Navarro, & D. Portthe (Eds.), *Proceedings of the 24th IEEE-CS Conference on Software Engineering Education and Training* (pp. 71-80). Honolulu, HI: IEEE Computer Society.

Barrett, M. S. (2012). Troubling the creative imaginary: Some possibilities of ecological thinking for music and learning. In D. Hargreaves, D. Miell, & R. Macdonald (Eds.), *Musical imaginations: Multidisciplinary perspectives on creativity, performance, and perception* (pp. 206-219). Oxford, England: Oxford University Press.

Barrett, M. S. (Ed.) (2010). *A cultural psychology of music education.* Oxford, England: Oxford University Press.

Bauer, W. I. (2012). The acquisition of musical technological pedagogical and content knowledge. *Journal of Music Teacher Education, 22*(2), 51-64.

Bauer, W. I. (2003). Gender differences and the computer self-efficacy of pre-service music teachers. *Journal of Technology in Music Learning, 2*(1), 9-15.

Bauer, W. I., & Dunn, R. E. (2003). Digital reflection: The electronic portfolio in music teacher education. *Journal of Music Teacher Education, 13*(1), 7-20.

Bauer, W. I., Reese, S., & McAllister, P. A. (2003). Transforming music teaching via technology: The role of professional development. *Journal of Research in Music Education, 51*(4), 289-301.

Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., & DeBaud, J. M. (1999). PuLSE: A methodology to develop software product lines. In the *Proceedings of the 5th ACM SIGSOFT Symposium on Software Reliability* (pp. 122-131). New York, NY: ACM Press.

Baylor, A. L., & Ritchie, D. (2002). What factors facilitate teacher skill, teacher morale, and perceived student learning in technology-using classrooms? *Computers & Education, 39*(4), 395-414.

Beckstead, D. (2001). Will technology transform music education? *Music Educators Journal, 87*(6), 44-49.

Berg, M., & Lind, R. (2003). Preservice music teacher electronic portfolios: Integrating reflection and technology. *Journal for Music Teacher Education, 12*(2), 18-29.

Bernard, H. R., Pelto, P., Romney, D., Ember, A., Johnson, A., Werner, O., Boster, J., et al. (1986). The construction of primary data in cultural anthropology. *Current Anthropology, 27*, 382-896.

Bernard, H. R., & Ryan, G. W. (2010). *Analyzing qualitative data: Systematic Approaches.* Thousand Oaks, CA: Sage.

Blakeslee, M., Brown, L. C., & Hofmann, A. O. (2008). *Model Music Programs: Ideas for Everyone.* Lanham, MD: Rowman & Littlefield Education.

Blombach, A. (2013). MacGAMUT (Version 6.1.9) [Software]. Gahanna, OH: MacGAMUT Music Software.

Boehm, C. (2005). Music technology in higher education. In F. McMahon, & T. Claes (Eds.), *Probing the Boundaries of Higher Education.* Oxford, England: Inter-Disciplinary Press.

Boehm, C. (2007). The discipline that never was: Current developments in music technology in higher education in Britain. *Journal of Music, Technology and Education, 1*(1), 7-12.

Borota, B., & Cencič, M. (2012). E-materials and computer usage within music education in Slovenian primary schools. In M. Gall, G. Sammer, & A. de Vugt (Eds.), *European perspectives on music education: New media in the classroom* (pp.205-218). Innsbruck, Australia: Helbling.

Bourque, P., Dupuis, R., Abran, A., Moore, J. W., & Tripp, L. (2004). *Guide to the software engineering body of knowledge.* Los Alamitos, CA, USA: IEEE Computer Society.

Boyce-Tillman, J. (2004). Towards an ecology of music education. *Philosophy of Music Education Review, 12*(2), 102–125.

Breeze, N. (2011). Multimodality: An illuminating approach to unraveling the complexities of composing with ICT? *Music Education Research, 13*(4), 389-405.

Britton, A. P., Bowles, E. A., Broido, A., Davis, C. W., Henney, T. H., Imig,

W., Sand, O., Shetler, D. J., & Whitney, M. C. (1968). Impact and potential of technology. In R. Choate (Ed.), *Documentary Report of the Tanglewood Symposium.* Washington, DC: Music Educators National Conference.

Brode, L., Zhou, H., Gibbons, A. (2008). Steps in developing an advanced software engineering course using problem based learning. *Engineering Education, 3*(1), 2-12.

Brown, A. R. (2007). *Computers in Music Education: Amplifying Musicality*. New York, NY: Routledge.

Bryman, A. (1988). *Quantity and quality in social research*. London, England: Routledge.

Buck, M. W. (2008). *The efficacy of SmartMusic assessment as a teaching and learning tool* (Unpublished doctoral dissertation). University of Southern Mississippi, Hattiesburg, MI.

Busen-Smith, M. (1999). Developing strategies for delivering music technology in secondary PGCE courses. *British Journal of Music Education 16*(2), 197-213.

Bush, J. (2001). Introducing the practitioner's voice through electronic mentoring. *Journal of Technology in Music Learning, 1*(1), 4-9.

Bushnell, D. D. (1962). Computer-based teaching machines. *The Journal of Education Research, 55*(9), 528-531.

Buxmann, P., Diefenbach, H., & Hess, T. (2013). *The software industry: Economic principles, strategies, perspectives.* Berlin, Germany: Springer.

Cain, T. (2004). Theory, technology and the music curriculum. *British Journal of Music Education, 21*(2), 215-221.

Cain, T. (2010). Using computers to teach listening skills: An intervention study. In J. Kerchner (Ed.), *18th Music in Schools and Teacher Education Seminar, International Society for Music Education* (pp. 13-17). Shenyang, China.

Callaghan, J., Thorpe, W., & van Doorn, J. (1999), Computer-assisted Visual Feedback in the Teaching of Singing. In M. S. Barrett, G. E. McPherson, & R. Smith (Eds.), *International Music Education Research Symposium: Vol.3. Children and Music: Developmental Perspectives* (pp. 105-111). Launceston, Tasmania.

Callaghan, J., Thorpe, W., & van Doorn, J. (2004). The Science of Singing and Seeing. In R. Parncutt, A. Kessler, & F. Zimmer (Eds.), *Conference on Interdisciplinary Musicology (CIM04)* (pp. 15-18). Austria: University of Graz.

Cambridge Dictionary Online (2012). Retrieved August 10, 2013, from http://dictionary.cambridge.org/

Campbell-Kelly, M. (1995). Development and structure of the international software industry, 1950-1990. *Business and Economic History, 24*(2), 73-110.

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An Empirical Study. *IEEE Software, 25*(1), 60-67.

Carroll, J. A., Potthoff, & D., Huber, T. (1996). Learning from three years of portfolio use in teacher education. *Journal of Teacher Education, 47,* 253-262.

Charmaz, K. (2004). Grounded theory. In S. Hesse-Biber & P. Leavy (Eds.), *Approaches to qualitative research: A reader on theory and practice* (pp. 496-521). New York, NY: Oxford University Press.

Chen, C. W. J. (2012). A pilot study mapping students' composing strategies: Implications for teaching computer-assisted composition. *Research Studies in Music Education, 34*(2), 157-171.

Cheng, L. (2010, March). *An algorithm for piano pedagogy through MIDI protocol and computer software.* Paper presented at the Centre for Information Technology in Education Research Symposium 2010 (CITERS 2010), the University of Hong Kong, Hong Kong, China.

Choksy, L., Abramson, R. M., Gillespie, A. E., Woods, D., & York, F. (2001). *Teaching Music in the Twenty-first Century* (2nd ed.). Upper Saddle River, NJ: Prentice-Hall.

Chong, E. K. M. (2010). The lure of Web 2.0 spaces. In M. Hannan (Ed.), *Proceedings of the 18th International Seminar of the Commission for the Education of the Professional Musician, International Society for Music Education* (pp. 91–102). Shanghai, China: Shanghai Conservatory of Music.

Chowning, J. (1973). The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society, 21*(7), 526-534.

Clements, P., & Northrop, L. M. (2002). *Software Product Lines Practices and Pattern.* Reading, MA: Addison-Wesley.

Collins, D. (2005). A synthesis process model of creative thinking in music composition. *Psychology of Music, 33*(2), 193-216.

Converse, J. M., & Presser, S. (1986). *Survey questions: Handcrafting the standardized questionnaire.* Beverly Hills, CA: Sage.

Cooper, A., Cronin, D., & Reimann, R. (2007). *About face 3: The essentials of interaction design.* New York, NY: Wiley.

Cox, K., Niazi, M., & Verner, J. (2009). Empirical study of Sommerville and Sawyer's requirements engineering practices. *IET Software, 3*(5), 339-355.

Creswell, J. (2004). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research.* Upper Saddle River, NJ: Prentice Hall.

Cusumano, M. A. (2004). *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad.* New York, NY: The Free Press.

Cysneiros, L. M., & Leite, J. C. S. D. (2004). Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering, 30*(5), 328-350.

Dammers, R. J. (2009). A survey of technology-based music classes in New Jersey high schools. *Contributions to Music Education, 36*(2), 25-43.

Dammers, R. J., & Bauer, W. I. (2012, November). *Technology in music teacher education: A national survey.* Paper presented at the Association for Technology in Music Instruction Conference, San Diego, CA.

Dardenne, A., van Lamsweerde, A., & Fickas, S. (1993). Goal-directed requirements acquisition. *Science of Computer Programming, 20*(1&2), 3-50.

Davis, N. (2010). Global interdisciplinary research into the diffusion of information technology innovations in education. In A. McDougall, & J. Murn (Eds.), *Researching I.T. in Education: Theory, practice and Future Directions.* London, England: Routledge.

Decker, B., Ras, E., Rech, J., Jaubert, P., & Rieth, M. (2007). Wiki-based stakeholder participation in requirements engineering. *IEEE Software, 24*(2), 28-35.

Dede, C. (1998). *Six challenges for educational technology.* Retrieved September 4, 2012, from http://www.virtual.gmu.edu/pdf/ASCD.pdf

Denzin, N. K., & Lincoln, Y. S. (2011). The discipline and practice of qualitative research. In N. K. Denzin, & Y. S. Lincoln (Eds.), *The Sage handbook of qualitative research* (pp. 1-19). Los Angeles, CA: Sage.

Docherty, M., Sutton, P., Brereton, M., Kaplan, S. (2001). An innovative design and studio-based CS degree. In the *Proceedings of the 32th SIGCSE Technical Symposium on Computer Science Education* (pp. 233-237). Charlotte, NC: ACM Press.

Dorfman, J. (2008). Technology in Ohio's school music programs: An exploratory study of teacher use and integration. *Contributions to Music Education, 35*(1), 23-46.

Dunbar-Hall, P, Rowley, J., Webb, M., & Bell, M. (2010). ePortfolios for music educators: Parameters, problems and possibilities. In W. Sims (Ed.), *Proceedings of the 29th World Conference of the International Society for Music Education* (pp. 219-223). Beijing, China.

Durlauf, S. N., & Young, H. P. (2001). *Social Dynamics.* Cambridge, MA: MIT Press.

Ebert, C. (2006). Four key requirements engineering techniques. *IEEE Software, 23*(3), 19-25.

Ebert, C., & Smouts, M. (2003). Tricks and traps of initiating a product line concept in existing products. In the *Proceedings of the 25th international Conference on Software Engineering* (pp. 520-525). Washington, DC: IEEE Computer Society.

Education and Manpower Bureau, Hong Kong (EMB). (1998a). *Information technology for qualify education: 5-Year strategy 1998/99 to 2002/03.* Hong Kong, China: The Government Printing Department.

Education and Manpower Bureau, Hong Kong (EMB). (1998b). *Information technology for learning in a new era: 5-Year strategy 1998/99 to 2002/03.* Hong Kong, China: The Government Printing Department.

Education and Manpower Bureau, Hong Kong (EMB). (2005). *Overall study on reviewing the progress and evaluating the Information Technology in Education (ITEd) Projects 1998/2003 - Final report.* Hong Kong, China: The Government Printing Department.

Evans, G., & Foord, J. (2008). Cultural mapping and sustainable communities: Planning for the arts revised. *Cultural Trends, 17*(2), 65.96.
Evans, S. (2011). Historical and comparative perspectives on the medium of instruction in Hong Kong. *Language Policy, 10*(1), 19-36.
Farzanfar, R. (2005). *Using qualitative research methods to evaluate automated health promotion/disease prevention technologies: A procedures' manual.* Boston, MA: Boston University Medical Campus.
Favaro, J. (2002). Managing requirements for business value. *IEEE Software, 19*(2), 15-17.
Finney, J. (2007). Music education as identity project in a world of electronic desires. In J. Finney, & P. Burnard (Eds.), *Music Education with Digital Technology*. London, England: Continuum Press.
Flanigan, G. P. (2008). *An investigation of the effects of the use of SmartMusic software by brass players on intonation and rhythmic accuracy* (Doctoral dissertation). Available from ProQuest Dissertations and Theses database. (AAT 3401785)
Flores, L. V., Vicari, R. M., & Pimenta, M. S. (2001). *Some heuristics for the development of music education software: First steps towards a methodology*. In *Proceedings of the 8th Brazilian Symposium on Computer Music*. Fortaleza, Brazil.
Fontana, A., & Frey, J. (2000). The interview: From structured questions to negotiated text. In N. K. Denzin, & Y. S. Lincoln (Eds.), *Handbook of qualitative research.* Thousand Oaks, CA: Sage.
Fowler, F. J. (2002). *Survey research methods.* London: Sage.
Fricker, S., Gorschek, T., Byman, C., & Schmidle, A. (2010). Handshaking with implementation proposals: Negotiating requirements understanding. *IEEE Software, 27*(2), 72-80.
Fritsch, C., & Hahn, R. (2004). Product line potential analysis. In *Proceedings of the Software Product Line Conference* (pp. 228-237).
Fung, V. (2003). Gender differences in preservice music educators' familiarity with technology. *Journal of Technology in Music Learning, 2*(1), 31-40.
Gall, M. (2013). Trainee teachers' perceptions: Factors that constrain the use of music technology in teaching placements. *Journal of Music, Technology & Education, 6*(1), 5-27.
Gall, M., de Vugt, A., & Sammer, G. (2012). Introduction to new media in the classroom. In M. Gall, G. Sammer, & A. de Vugt (Eds.), *European perspectives on music education: New media in the classroom* (pp.11-29). Innsbruck, Australia: Helbling.
Gall, M., & Breeze, N. (2007). The sub-culture of music and ICT in the classroom. *Pedagogy and Education, 16*(1), 41-56.
Gall, M., & Breeze, N. (2005). Music composition lessons: The multimodal affordances of technology. *Educational Review, 57*(4), 415-433.
Gannod, G. C., Burge, J. E., Helmick, M. T. (2008). Using the inverted classroom to teach software engineering. In *Proceedings of the 30th International Conference on Software Engineering* (pp. 777-786). Leipzig, Germany: ACM Press.

Garlan, D., Gluch, D. P., Tomayko, J. E. (1997). Agents of change: Educating software engineering leaders. *IEEE Computer, 30*(11), 59-65.

Gary, K., & Varma, V. (2007). A study of the effectiveness of case study approach in software engineering education. In *Proceedings of the 20th Conference on Software Engineering Education and Training* (pp. 309-316). Dublin, Ireland: IEEE Computer Society.

Gast, H. (2008). Patterns and traceability in teaching software architecture. In the *Proceedings of the 6th International Symposium on Principles and Practice of Programming in Java* (pp. 23-31). Modena, Italy: ACM Press.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research.* Chicago, IL: Aldine.

Glenn, S. G. (2000). *The effects of a situated approach to musical performance education on student achievement: Practicing with an artificially intelligent computer accompanist* (Doctoral dissertation). Available from ProQuest Dissertations and Theses database. (AAT 9984138)

Gilb, T. (2005). *Competitive engineering: A handbook for systems engineering, requirements engineering, and software Engineering using Planguage.* Amsterdam, Netherlands: Elsevier.

Greher, G. R. (2004). Multimedia in the classroom: Tapping into an adolescent's cultural literacy. *Journal of Technology in Music Learning, 2*(2), 21-43.

Greher, G. R. (2011). Music technology partnerships: A context for music teacher preparation. *Arts Education Policy Review, 112*(3), 130-136.

Goodlad, J. I. (1985). Rethinking what schools can do the best. In E. W. Eisner (Ed.), *National Society for the Study of Education, 84th Yearbook* (pp. 29-45). Chicago IL: University of Chicago Press.

Gottesdiener, E. (2002). *Requirements by Collaboration: Workshops for Defining Needs.* Boston, MA: Addison-Wesley.

Gottesdiener, E. (2003). Requirements by collaboration: Getting it right the first time. *IEEE Software, 20*(2), 52-55.

Gouzouasis, P. (2005). Fluency in general music and arts technologies: Is the future of music a garage band mentality? *Action, Criticism and Theory for Music Education, 4*(2).

Greetz, C. (1973). *The interpretation of cultures.* New York, NY: Basic Books.

Grega, W., Kornecki, A. J., Sveda, M., & Thiriet, J-M. (2007). Developing an interdisciplinary and multinational software engineering curriculum. In the *Proceedings of International Conference on Engineering Education.* Coimbra, Portugal: International Network on Engineering Education and Research.

Halfpenny, P. (1979). The analysis of qualitative data. *Sociological Review, 27*(4), 799.825.

Hall, C. V., & O'Donnell, J. (2010). Calibrating a bowing checker for violin students. *Journal of Music, Technology & Education, 3*(2&3), 125-139.

Hammersley, M. (1992). *What's wrong with ethnography: Methodological explorations.* London, England: Routlege.

Hammersley, M. (1990). *Reading ethnographic research: A critical guide.* London, England: Longmans.

Han, J. (2008). The business strategy of McDonald's. *International Journal of Business and Management, 3*(11), 72-74.
Harris, M. E., Tharp, T., & Zalewski, J. (2002, June). *Internationalization of software engineering courses: First experiences and future direction.* Paper presented at the 6th Biennial World Conference on Integrated Design & Process Technology, Pasadena, CA. Abstract available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.9873&rep=rep1&type=pdf
Haynes, C. (2002). *Innovations in interdisciplinary teaching.* Westport, CT: Oryx Press.
Hazzan, O. (2002). The reflective practitioner perspective in software engineering education. *Journal of Systems and Software, 63*(3), 161-171.
Heaven, W., & Finkelstein, A. (2004). UML profile to support requirements engineering with KAOS. *IEEE Proceedings – Software, 151*(1), 10-27.
Hesse-Biber, S. N., & Leavy, P. (2004). *Approaches to Qualitative Research: A reader on theory and practice.* New York, NY: Oxford University Press.
Hesse-Biber, S. N., & Leavy, P. (2011). *The practice of qualitative research* (2nd ed.). Thousand Oaks, CA: Sage.
Hilburn, T. B., Towhidnejad, M., Nangia, S., & Shen, L. (2006). A case study project for software engineering education. In *Proceedings of the 36th ASEE/IEEE Frontiers in Education Conference* (pp. 1-5). San Diego, CA: IEEE Computer Society.
HKCDC (2003). *Arts education key learning area: Music curriculum guide (Primary 1 – Secondary 3)*. Hong Kong, China: Government Logistics Department.
HKCDC & HKEAA (2007). *Music curriculum and assessment guide (Secondary 4–6)*. Hong Kong, China: Government Logistics Department.
Hoch, D. J., Roeding, C. R., Purkert, G., Lindner, S. K., & Mueller, R. (1999). *Secrets of software success: Management insights from 100 software firms around the world.* Boston, MA: Harvard Business School Press.
Hodges, R. (2001). Using ICT in music teaching. In C. Philpott, & C. Plummeridge (Eds.), *Issues in music teaching* (pp. 170–181). London & New York: Routledge.
Hodges, R. (2007). Music education and training: ICT, Innovation and curriculum reform. In J. Finney, & P. Burnard (Eds.), *Music Education with digital technology* (pp. 169-180). London: Continuum Press.
Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Software, 18*(4), 4-66.
Hofstetter, F. T. (1981). Applications of the GUIDO system to aural skills research. *College Music Symposium, 21*, 46–53.
Hofstetter, F. T. (1980). Computer-based recognition of perceptual patterns in chord quality dictation exercises. *Journal of Research in Music Education, 28*(2), 83-91.
Hofstetter, F. T. (1979). Evaluation of a competency-based approach to teaching aural interval identification. *Journal of Research in Music Education, 27*(4), 201-213.

Hofstetter, F. T. (1978). Computer-based recognition of perceptual patterns in harmonic dictation exercises. *Journal of Research in Music Education, 26*(2), 111-119.

Hogan, J. M., Smith, G., & Thomas, R. (2005). Tight spirals and industry clients: The modern SE education experience. In the *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42* (pp. 217-222). Newcastle, New South Wales, Australia: Australian Computer Society.

Holmes, T. (2002). *Electronic and experimental music: Pioneers in technology and composition* (2nd ed.). New York & London: Routledge.

Hopkins, M. (2002). The effects of computer-based expository and discovery methods of instruction on aural recognition of music concepts. *Journal of Research in Music Education, 50*(2), 131-144.

Ho, W. C. (2007). Music students' perception of the use of multi-media technology at the graduate level in Hong Kong higher education. *Asia Pacific Education Review, 8*(1), 12-26.

Ho, W. C. (2009). The role of multimedia technology in a Hong Kong higher education music programs. *Vision of Research in Music Education, 13*. Retrieved June 24, 2011, from http://www-usr.rider.edu/~vrme/

Howard, D. M., Welch, G. F., Brereton, J., Himonides, E., & Howard, A. W. (2004). WinSingad: A real-time display for the singing studio. *Logopedics Phoniatrics Vocology, 29*(3), 135-144.

Huang, S., & Distante, D. (2006). On practice-oriented software engineering education. In the *Proceedings of the 19th Conference on Software Engineering Education and Training Workshops* (pp. 15-18). Washington, DC: IEEE Computer Society.

Jaakkola, H. (2009). Towards a globalized software industry. *Acta Polytechnica Hungarica, 6*(5), 69-84.

Jackson, K. M., & Trochim, W. M. K. (2002). Concept mapping as an alternative approach for the analysis of open-ended survey responses. *Organizational Research Methods, 5*(4), 307-336.

Jassmann, A. (2004). *The status of music technology in the K-12 curriculum of South Dakota public schools.* Dissertation Abstracts International, 65 (04), 1294. (UMI No. 3127829).

Jautakyte, Z. (2012). ICT in music: A case of Lithuanian upper school education. In M. Gall, G. Sammer, & A. de Vugt (Eds.), *European perspectives on music education: New media in the classroom* (pp.149-162). Innsbruck, Australia: Helbling.

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Towards a definition of mixed method research. *Journal of Mixed Methods Research, 1*(2), 112-133.

Jones, C. (1996). *Applied software measurement: Assuring productivity and quality.* New York, NY: McGraw-Hill.

Kang, K. C., Donohoe, P., Koh, E., Lee, J., & Lee, K. (2002). Using a marketing and product plan as a key driver for product line asset development. In the *Proceedings of the 2nd International Conference of Software Product Lines* (pp. 366-382). San Diego, CA.

Käkölä, T. (2003). Software business models and contexts for software innovation: Key areas for software business research. In the *Proceedings of the 36th Hawaii International Conference on System Sciences* (HICSS'03) (pp. 1-8). Big Island, HI.

Keeling, K., Keeling, D., & McGoldrick, P. (2011). Retail relationships in a digital age. *Journal of Business Research.* Retrieved November 12, 2012, from http://www.sciencedirect.com/science/article/pii/S0148296311001974

Kelly, A. V. (2009). *The Curriculum: Theory and Practice* (6th ed.). London, England: Sage.

Khmelevsky, Y. (2009). SW development projects in academia. In the *Proceedings of the 14th Western Canadian Conference on Computing Education* (pp. 60-64). Burnaby, British Columbia, Canada: ACM Press.

Kirkman, P. R. (2010). Exploring contexts for development: Secondary music students' computer-mediated composing. *Journal of Music, Technology & Education, 3*(2&3), 107-124.

Kokkoniemi, J. K. (2008). Gathering experience knowledge from iterative software development processes. In the *Proceedings of the 41st Annual Hawaii International Conference on System Sciences* (pp.333). Waikoloa, HI: IEEE Computer Society.

Koller, V. (2007). "The World's Local Bank": Glocalisation as a strategy in corporate branding discourse. *Social Semiotics, 17*(1), 111-131.

Kontio, J., Ahokas, M., Pöyry, P., Warsta, J., Mäkelä, M., & Tyrväinen, P. (2006). Software business education for software engineers: Towards an integrated curriculum. In D. Port, & L. Williams (Eds.), *Proceedings of 19th Conference on Software Engineering Education and Training Workshops* (CSEE&T) (pp. 5-8). Oahu, HI: IEEE Computer Society.

Krause, S. (1996). Portfolios in teacher education: Effects of instruction on preservice teachers' early comprehension of the portfolio process. *Journal of Teacher Education, 47,* 130-38.

Kruse, N. B., Harlos, S. C., Callahan, R. M., & Herring, M. L. (2013). Skype music lessons in the academy: Intersections of music education, applied music and technology. *Journal of Music, Technology & Education, 6*(1), 43-60.

Krüger, S. E., Fritsch, E. F., Flores, L. V., Grandi, R. H., Santos, T. R., Hentschke, L., & Viccari, R. M. (1999). Developing a software for music education: An interdisciplinary project. In *Proceedings of the 19th Congresso Nacional da Sociedade Brasileira de Computação* (pp.251-264). Rio de Janeiro, Brazil: EntreLugar.

Kuhn, S. (1998). The software design studio: An exploration. *IEEE Software, 15*(2), 65-71.

Laplante, P. A. (2007). *What every engineer should know about software engineering*. Boca Raton, FL: CRC Press.

Largillier, G. (2007). Developing the first commercial product that uses multi-touch technology. *Information Display, 23*(12), 14-10.

Lauesen, S. (2003). Task descriptions as functional requirements. *IEEE Software, 20*(2), 58-65.

Lawrence, R. J. (2010). Beyond disciplinary confinement to imaginative transdisciplinarity. In V. A. Brown, J. A. Harris, & J. Y. Russell (Eds.), *Tackling wicked problems: Through the transdisciplinary imagination* (pp. 16-30). London, England: Routledge.

Liao, Z., & Shi, X. (2009). Emerald Article: Consumer perceptions of internet-based e-retailing: An empirical research in Hong Kong. *Journal of Services Marketing, 23*(1), 24-30.

Louridas, P. (2006). Using wikis in software development. *IEEE Software, 23*(2), 88-91.

Lee, B. K. (2010). ICT integration in primary school music education: Experience of pioneering countries and its implications for implementation in Hong Kong. *Asia-Pacific Journal for Arts Education, 8*(4), 1-30.

Lee, D., & Gilmore, A. (2012). Mapping cultural assets and evaluating significance: Theory methodology and practice. *Cultural Trends, 21*(1), 3-28

Lee, E. (2007). *A study of the effect of computer assisted instruction, previous music experience, and time on the performance ability of beginning instrumental music students* (Doctoral dissertation). Available from ProQuest Dissertation and Theses database. (AAT 3284028)

Lethbridge, T. C., Díaz-Herrera, J., LeBlanc, R. J., Jr., & Thompson, J. B. (2007). Improving software practice through education: Challenges and future trends. In the *Proceedings of Future of Software Engineering* (FOSE '07). Minneapolis, MN.

Lenberg, H. (2010). Music software for the masses (Master's thesis, Royal Institute of Technology, Stockholm, Sweden). Retrieved from http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2010/sammanf10/lenberg_henrik.html

Leong, S. (2002). Interdisciplinary teamwork in developing metacognitive software for melodic dictation: Lessons from an Australian National Teaching Development project. In M. Espeland (Ed.), *Proceedings of the 25th World Conference of the International Society for Music Education* (pp. 219-223). Bergen, Norway: International Society of Music Education.

Leong, S. (2011). Navigating the emerging futures in music education. *Journal of Music, Technology and Education, 4*(2&3), 233-243.

Lethbridge, T. C. (2000). What knowledge is important to a software professional? *IEEE Computer, 33*(5), 44-50.

Leung, B. W. (2003). Teaching musicianship in Hong Kong: Current issues and future trends. In S. Leong (Ed.), *Musicianship in the 21st century: Issues, trends and possibilities* (pp. 17-.185). Marrickville, Australia: Southwood Press.

van der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., & Obbink, H. (2004). Software product family evaluation. In the *Proceedings of the 3rd International Conference on Software Product Lines* (pp. 110-129). Boston, MA.

Long, M. K. (2011). *The effectiveness of the SmartMusic assessment tool for evaluating trombone student performance* (Unpublished doctoral dissertation). University of North Carolina at Greensboro, Greensboro, NC.

Löwgren, J., & Stolterman, E. (2004). *Design av informationsteknik: Materialet utan egenskaper.* Lund, Sweden: Studentlitteratur AB.
Lucena, J. (2006). Globalization and organizational change: Engineers' experiences and their and their implications for engineering education. *European Journal of Engineering Education, 31*(3), 321-338.
Lum, C, H. (2008). Home musical environment of children in Singapore on globalization, technology and media. *Journal of Research in Music Education, 56*(2), 101-117.
MacGAMUT Music Software. (2013). *About MacGAMUT.* Retrieved June 24, 2013, from http://www.macgamut.com/about/
Maedche, A., Botzenhardt, A., & Neer, L. (2012). Software for people: A paradigm change in the software industry. In A. Maedche, A. Botzenhardt, & L. Neer (Eds.), *Software for people* (pp. 1-8). Berlin, Germany: Springer.
Maiden, N. (2006). Improve your requirements: Quantify them. *IEEE Software, 23*(6), 68-69.
Maiden, N. (2007). My requirements? Well, that depends. *IEEE Software, 24*(1), 86-87.
Maiden, N., & Jones, S. (2010). Agile requirements – Can we have our cake and eat it too? *IEEE Software, 27*(3), 87-88.
Mäkelä, M. (2005, May). *Software business: Position as a field of research and avenues for scholarly contributions.* Paper presented at the 14th International Conference of the International Association for Management of Technology (IAMOT), Vienna, Austria.
Mäkelä, M., & Mutanen, O. (2005). Research in software business: Implications of the special qualities of software as a good. In the *Proceedings of Engineering Management Conference, Vol. 2* (pp. 780-783). IEEE International.
Manning, P. (2013). *Electronic and Computer Music* (4th ed.). Oxford: Oxford University Press.
Manyika, J. M., Roberts, R. P., & Sprague, K. L. (2007). Eight business technology trends to watch. *The McKinsey Quarterly*, December 2007.
Manzolli, J. & Verschure, P. F. M. J. (2005). Roboser: A real-world composition system. *Computer Music Journal, 29*(3), 55-74.
Mark, M. L. (1986). Materials and tools of music education. In M. L. Mark (Ed.), *Contemporary music education* (pp. 203-218). New York, NY: Schirmer Books.
Marciniak, J. J. (1994). *Encyclopedia of Software Engineering.* New York, NY: Wiley.
McKinney, M. (1998). Preservice teachers' electronic portfolios: Integrating technology, self-assessment, and reflection. *Teacher Education Quarterly, 25*(1), 85-103.
McLuhan, M. (1964). *Understanding Media: The Extension of Man.* London, England: Sphere Books.
Meltzer, J. (2001). *A survey to assess the technology literacy of undergraduate music majors at Big-10 universities: Implications for undergraduate*

courses in music education technology. Unpublished doctoral dissertation, University of Illinois at Urbana – Champaign.

Miles, T. (2010). *Software and IT services industry data* [PowerPoint slides]. Washington, DC: Office of Technology and Electronic Commerce, Manufacturing and Services, Department of Commerce. Retrieved August 27, 2013, from: http://www.commerce.gov/

Montague, J. (2010). Challenges for distance education: An online music classroom. In C. C. Leung, L. C. Yip, & T. Imada (Eds.), *Music Education Policy and Implementation: Culture and Technology* (pp. 93-97). Henan, China: Henan University.

Moore, J. F. (1996). *The Death of competition: Leadership & strategy in the age of business ecosystems.* New York, NY: HarperBusiness.

Myllykoski, M. (2012). The use of ICT and music technology in Finnish music education. In M. Gall, G. Sammer, & A. de Vugt (Eds.), *European perspectives on music education: New media in the classroom* (pp.115-124). Innsbruck, Australia: Helbling.

Mylopoulos, J. (1992). Conceptual modeling and Telos. In P. Loucopoulos, & R. Zicari (Eds.), *Conceptual modeling, databases, and case: An integrated view of information systems development* (pp. 49-68). New York, NY: Wiley.

Nam, N., & Harter, D. E. (2009). Impact of budget and schedule pressure on software development cycle time and effort. *IEEE Transactions on Software Engineering, 35*(5), 624-637.

Naru, P., & Randell, B. (1968). Software engineering and society. In P. Naru & B. Randell (Eds.), *Proceedings of NATO Software Engineering Conference, 1,* 19-34.

Naughton, C. (1997). Music technology tools and the implications of socio-cognitive research. *British Journal of Music Education, 14*(2), 111-117.

Neill, C. J., & Laplante, P. A. (2003). Requirements engineering: The state of the practice. *IEEE Software, 20*(6), 40-45.

Newcomb, S. R., Weage, B. K., & Spencer, P. (1981). "MEDICI" tutorial in melodic dictation. *Journal of Computer-based Instruction, 7*(3), 63-69.

Newell, W. H. (2007). Decision making in interdisciplinary studies. In G. Morçöl (Ed.), *Handbook of decision making* (pp. 245-265). New York: Marcel Dekker.

Nielson, J. M. (Ed.) (1990). Introduction. *Feminist among methods* (pp. 1-37). Boulder, CO: Westview Press.

Nilsson, B., & Folkestad, G. (2005). Children's practice of computer-based composition. *Music Education Research, 7*(1), 21-37.

Ng, K. (2008). Technology-enhanced learning for music with i-Maestro framework and tools. In S. Dunn, S. Keene, G. Mallen, & J. Bowen (Eds.), *Proceedings of the EVA 2008 London International Conference on Electronic Visualisation and the Arts* (pp. 177-187). London, England: British Computer Society.

Noble, J., Marshall, S., Marshall, S., & Biddle, R. (2004). Less extreme programming. In the *Proceedings of the 6th Conference on Australasian*

Computing Education – Volume 30 (pp. 217-226). Dunedin, New Zealand: Australian Computer Society.

Noxon, J. (2003). Music technology as a team sport. *Journal of Technology in Music Learning, 2*(2), 56-61.

Ofsted. (2009). *Making more of music: An evaluation of music in schools 2005/2008 (080235).* London, England: Ofsted.

Ofsted. (2012). *Music in schools: Wider still, and wider (110158).* Manchester, England: Ofsted.

Ohlenbusch, G. (2001). *A study of the use of technology applications by Texas music educators and the relevance to undergraduate music education curriculum.* Unpublished doctoral dissertation, Shenandoah Conservatory, Winchester, VA.

Opler, M. E. (1945). Themes as dynamic forces in culture. *American Journal of Sociology, 51*, 198-206.

Padgett, D., & Mulvey, M. S. (2007). Differentiation via technology: Strategic positioning of services following the introduction of disruptive technology. *Journal of Retailing, 83*(4), 375-391.

Palmer, A. J., & de Quadros, A. (2012). A new dream. In A. J. Palmer & A. de Quadros (Eds.), *Tanglewood II: Summoning the future of music education* (pp.17-24). Chicago, IL: GIA Publications.

Parker, R. W. (1974). *The relative effectiveness of the TAP system in instruction in sight-reading: An experimental study* (Unpublished doctoral dissertation). University of Miami, Coral Gables, FL.

Parnas, D. (1999). Education for computing professionals. *IEEE Computer, 23*(1), 17-22.

Partti, H. (in press). Cosmopolitan musicianship under construction: Digital musicians illuminating emerging values in music education. *International Journal of Music Education.*

Peters, G. D. (1979). *Courseware development for microcomputer-based instruction in music.* Paper presented at the Association for the Development of Computer-based Instruction Systems, San Diego, CA.

Perkmen, S., & Cevik, B. (2010). Relationship between pre-service music teachers' personality and motivation for computer-assisted instruction. *Music Education Research, 12*(4), 415.425.

Price, H., & Pan, K. (2002). A survey of music education technology at colleges in the southeastern USA. *Journal of Technology of Music Education, 19*(1), 61-71.

Popp, K. M., & Meyer, R. (2010). *Profit from software ecosystems: Business models, ecosystems and partnerships in the software industry.* Norderstedt, Germany: Books on Demand.

Rajagopa, P., Lee, R., Ahlswede, T., Chiang, C., & Karolak. D. (2005). A new approach for software requirements elicitation. In the *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks* (pp. 32-42). Washington, DC: IEEE Computer Society.

Rapanotti, L., Hall, J. G., & Li, Z. (2006). Deriving specifications from requirements through problem reduction. *IEEE Proceedings – Software, 153*(5), 183-198.

Ranjan, P., & Misra, A. K. (2009). A novel approach of requirement gathering and analysis for agent oriented software engineering (AOSE). *International Journal of Software Engineering and Knowledge Engineering, 19(*1), 79-111.

Rees, F. (2002). Distance learning and collaboration in music education. In R. Colwell, & C. Richardson (Eds.), *The new handbook of research on music teaching and learning* (pp.257-273). New York, NY: Oxford.

Reese, S., & Rimington, J. (2000). Music technology in Illinois public schools. *Update: Applications of Research in Music Education, 18*(2), 27-32.

Rico, D. F., & Sayani, H. H. (2009). Use of agile methods in software engineering education. In the *Proceedings of the 2009 Agile Conference* (pp.174-179). Chicago, IL: IEEE Computer Society.

Roblyer, M. D. (2006). *Integrating Educational Technology into Teaching* (4th ed.). Upper Saddle River, NJ: Pearson.

Robertson, S. (2004). Requirements and the business case. *IEEE Software, 21*(5), 93-95.

Robertson, S., & Robertson, J. (1999). *Mastering the Requirement Process.* Boston, MA: Addison-Wesley.

Rönkkö, M., Valtakoski, A., Peltonen, J. (2010). The case for software business as a research discipline. In P. Tyrväinen, S. Jansen, & M. A. Cusumano (Eds.), *Proceedings of the First International Conference on software business, Lecture Notes in Business Information Processing, Vol. 51* (pp. 205-210). Jyväskylä, Finland: Springer Berlin Heidelberg.

Salavuo, M. (2006). Open and informal online communities as forums of collaborative musical activities learning. *British Journal of Music Education, 23*(3), 253-271.

Sandelowski, M. (1995). Qualitative analysis: What is and how to begin. *Research in Nursing and Health, 18*, 371-375.

Santos, S. C., Batista, M. da C. M., Cavalcanti, A. P. C., Albuquerque, J. O., & Meira, S. R. de L. (2009). Applying PBL in software engineering education. In *Proceedings of the 22nd Conference on Software Engineering Education and Training* (pp. 192-189). IEEE Computer Society.

Savage, J. (2010). A survey of ICT usage across English secondary schools. *Music Education Research, 12*(1), 47-62.

Savage, J. (2007). Reconstructing music education through ICT. *Research in Education, 78*, 65-77.

Savage, J. (2005, March 31). Information communication technologies as a tool for re-imagining music education in the 21th century. *International Journal of Education & the Arts, 6*(2). Retrieved June 24, 2011, from http://www.ijea.org/v6n2/

Savage, J., & Challis, M. (2002). A digital arts curriculum? Practical ways forward. *Music Education Research, 4*(1), 7-24.

Scott, A. (1968). A study of the components of the singing tone utilizing the audio spectrum analyzer. *The NATS Bulletin, 57*(5), 40-41.
Schlager, K. (2008). Distance learning. *Teaching Music, 15*(6), 36-38.
Schön, D. A.(1987). *Educating the reflective practitioner: Towards a new design for teaching and learning in the profession.* San Francisco, CA: Jossey-Bass.
Schön, D. A.(1983). *The Reflective Practitioner.* London, England: Basic Books.
Schutz, A. (1967). The Phenomenology of the social world. Evanston, IL: Northwestern University Press.
Schwartz, D. (2008). CLOWNS - A software engineering semester project currently in-use in kindergarten classes. In H. R. Arabnia, V. A. Clincy, & N. Tadayon (Eds.), the *Proceedings of the 2008 International Conference on Frontiers in Education: Computer Science & Computer Engineering* (pp. 344-349). Las Vegas, NV: CSREA Press.
Sebern, M. J. (1997). Iterative development and commercial tools in an undergraduate software engineering course. In the *Proceedings of the 28th SIGCSE technical Symposium on Computer Science Education* (pp. 306-309). San Jose, CA: ACM Press.
Seddon, F. A. (2007). Music e-learning environments: Young people, composing and the Internet. In J. Finney, & P. Burnard (Eds.), *Music education with digital technology*. London, England: Continuum Press.
Seddon, F. A. (2002). *An interpretation of composition strategies adopted by adolescents with and without prior experience of formal instrumental music tuition (FIMT) engaging with computer-based composition.* Paper presented at the 25th International Society of Music Education Conference, Bergen, Norway.
Seddon, F. A., & O'Neill, S. (2003). Creative thinking processes in adolescent computer-based compositions: An analysis of strategies adopted and the influence of instrumental music training. *Music Education Research, 5*(2), 125-135.
Shaw, M. (2000). Software engineering education: A roadmap. In A. Finkelstein (Ed.), *Proceedings of the Conference on the Future of Software Engineering* (pp. 371-380). New York, NY: ACM Press.
Sidwell-Frame, K. (2009). *The effect of SmartMusic on the attitudes and achievement of 5th grade students* (Unpublished master's thesis). Viterbo University, La Crosse, WI.
Silverman, D. (1985). *Qualitative methodology and sociology*. Aldershot, England: Gower.
Skeldon, K. D., Reid, L. M., McInally, V., Dougan, B., Fulton, C. (1988). Physics of the Theremin. *American Journal of Physics, 66*(11), 945-955.
Smith, T. M., & Smith, R. L. (2012). *Elements of Ecology* (8th ed.). Boston, MA: Benjamin Cummings.
Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering – A Good Practice Guide.* New York, NY: John Wiley & Sons.

Southcott, J., & Crawford, R. (2011). The intersections of curriculum development: Music, ICT and Australian music education. *Australasian Journal of Educational Technology, 27*(1), 122-136.
Spangler, D. R. (1999). *Computer-assisted instruction in ear-training and its integration into undergraduate music programs during the 1998-1999 academic year.* Unpublished master's thesis, Michigan State University.
Stake, R. E. (1995). *The art of case research.* Thousand Oaks, CA: Sage.
The Standish Group (1994). *The CHAOS Report.* Retrieved July 17, 2011, from http://www.standishgroup.com/sample_research/chaos_1994_1.php
Stone, A., & Sawyer, P. (2006). Identifying tacit knowledge-based requirements. *IEEE Proceedings – Software, 153*(6), 221-218.
Strauss, A. L., & Corbin, J. (1990). *Basics of qualitative research: Grounded theory procedures and techniques.* Newbury Park, CA: Sage.
Sutherland, M., & Maiden, N. (2010). Storyboarding Requirements. *IEEE Software, 27*(6), 9-11.
Svensson, G. (2001). "Glocalization" of business activities: A "glocal strategy" approach". *Management Decision, 39*(1), 6-18.
Taylor, J. A. (2003). The status of technology in K-12 music education. *Journal of Technology in Music Learning, 2*(2), 67-73.
Taylor, J. A., & Deal, J. J. (2003). *The Status of Technology Integration in College Music Methods Courses: A survey of NASM Colleges and Universities.* Paper presented at the meeting of the Association for Technology in Music Instruction, Santa Fe, NM.
Tappeiner, G., Tappeiner, U., & Walde, J. (2007). Integrating disciplinary research into an interdisciplinary framework: A case study in sustainability research: Introduction to the special issue. *Environmental Modeling and Assessment, 12*(4), 253-256.
Thornton, L., Ferris, N. Johnson, G., Kidwai, K., Ching, Y-H. (2011). The impact of an e-portfolio program in a music education curriculum. *Journal of Music Teacher Education, 21*(1), 65-77.
Thorgersen, K. (2010). *Democracy, open source and music education? A Deweyan investigation of music education in digital domains.* Malmö: Sweden.
Thorpe, W. (2002). Visual feedback of acoustic voice features in voice training. In *9th Australian Speech Science & Technology Conference* (pp.349-354). Australia: University of Melbourne.
Thorpe, W. Callaghan, J., Wilson, P., van Doorn, J., & Crane, J. (2013). Sing & See (Version 1) [Software]. Auckland, New Zealand: CantOvation.
Tobias, E. S. (2012). Hybrid spaces and hyphenated musicians: Secondary students' musical engagement in a songwriting and technology course. *Music Education Research, 14*(3), 329-346.
Tomayko, J. E. (1996). Carnegie-Mellon's software development studio: A five-year retrospective. In the *Proceedings of the 9th Conference on Software Engineering Education* (pp.119-129). Los Alamitos, CA: IEEE Computer Society Press.
Tomayko, J. E. (1991). Teaching software development in a studio environment. *ACM SIGCSE Bulletin, 23*(1), 300-303.

Toft, P., Coleman, D., & Ohta, J. (2000). A cooperative model for cross-divisional product development for a software product line. In the *Proceedings of the 1st International Conference on Software Product Lines* (pp. 111-132). Denver, CO.

Tsui, F., & Karam, O. (2011). *Essentials of software engineering* (2nd ed.). Sunbury, MA: Jones & Bartlett Learning.

Upitis, R. (2001). Spheres of influence: The interplay between music research, technology, heritage, and music education. *International Journal of Music Education, 37*, 44-58.

Valtakosi, A., & Rönkkö,M. (2009). *The concept of business model in management practice and research: Bridging the relevance gap.* Paper presented at the Academy of Management Annual Meeting in Chicago, Helsinki University of Technology, Espoo, Finland.

Van Vliet, H. (2008). Introduction. *Software engineering: Principles and practice* (3rd ed.). Chichester, UK: John Wiley & Sons.

Van Vliet, H. (2006). Reflections on software engineering education. *IEEE Software, 23*(3), 55-61.

Waldron, J. (2009). Exploring a virtual music 'community of practice': Informal music learning on the Internet. *Journal of Music, Technology & Education, 2*(2&3), 97-112.

Wanger, E. (1998). *Communities of practice: Learning, meaning, and doing.* New York, NY: Cambridge University Press.

Ward, C. J. (2009). Musical exploration using ICT in the middle and secondary school classroom. *International Journal of Music Education, 27*(2), 154-168.

Webster, P. R. (2011a). Constructivism and music learning. In R. Colwell, & P. Webster (Eds.), *MENC Handbook of Research on Music Learning, Vol. 1* (pp. 35–83). New York, NY: Oxford University.

Webster, P. R. (2011b). Key research in music technology and music teaching and learning. *Journal of Music, Technology and Education, 4*(2&3), 115-130.

Webster, P. R. (2007). Computer based technology and music teaching and learning: 2000-2005. In L. Bresler (Ed.), *International handbook of research in arts education* (pp. 1311-1328). Dordrecht, Netherlands: Springer.

Webster, J. (2000). Engineering education in Australia. *International Journal of Engineering Education, 16*(2), 146-153.

Welch, G. F. (2010). Ecological validity and impact: Key challenges for music education research. In T. A. Regelski, & J. T. Gates (Eds.), *Music education for changing times, landscapes: The arts, aesthetics, and education* (pp. 149-159). Netherlands: Springer.

Welch, G. F., Himonides, E., Howard, D. M., & Brereton, J. (2004). VOXed: Technology as a meaningful teaching aid in the singing studio. In R. Parncutt, A. Kessler, & F. Zimmer (Eds.), *Proceedings of the 4th Conference on Interdisciplinary Musicology*. Austria: University of Graz.

Welch, G. F., Howard, D. M., Himonides, E., & Brereton, J. (2005). Real-time feedback in the singing studio: An innovatory action-research project using new voice technology. *Music Education Research, 7*(2), 225-249.
Welch, G. F., Purves, R., Hargreaves, D., & Marshall, N. (2011). Early career challenges in secondary school music teaching. *British Educational Research Journal, 37*(2), 285-315.
Wiegers, K. E. (2003). *Software Requirements* (2nd ed.). Redmond, WA: Microsoft Press.
Williams, D. B., & Webster, P. R. (2008). *Experiencing music technology* (3rd ed.). Boston, MA: Schirmer Cengage Learning.
Williams, D. B., & Shrader, D. L. (1980). *The development of a microcomputer-based music instruction lab.* Paper presented at the NCCBMI/ADCIS National Conference, Arlington, VA.
Wilson, P. H., Lee, K., Callaghan, J., & Thorpe, C. W. (2007). Learning to sing in tune: Does real-time visual feedback help? In K. Maimets-Volk, R. Parncutt, M. Marin & J. Ross (Eds.), Proceedings of the 7th *Conference on Interdisciplinary Musicology* (pp. 157-172). Tallinn, Estonia.
Wilson, P. H., Lee, K., Callaghan, J. & Thorpe, C. W. (2008). Learning to sing in tune: does real-time visual feedback help? *Journal of Interdisciplinary Music Studies*, *2*(1&2), 157-172.
Wilson, P. H., Thorpe, C. W., & Callaghan, J. (2005). Looking at singing: Does real-time visual feedback improve the way we learn to sing? In *2nd APSCOM Conference: Asia-Pacific Society for the Cognitive Sciences of Music*. Seoul, South Korea: Asia-Pacific Society for the Cognitive Sciences of Music.
Winterson, J., & Russ, M. (2009). Understanding the transition from school to university in music and music technology. *Arts & Humanities in Higher Education, 8*(3), 339-354.
Wise, S. (2010). Teachers' perceptions of the impact of ICT in secondary music education. In W Sims (Ed.), *Proceedings of the 29th World Conference of the International Society for Music Education* (pp. 219-223). Beijing, China.
Wise, S., Greenwood, J., & Davis, N. (2011). Teachers' use of digital technology in secondary music education: Illustrations of changing classrooms. *British Journal of Music Education, 28*(2), 117-134.
Wu, W. H., Chen, W. F., Wang, T. L., and Su, C. H. (2008). Developing and evaluating a game-based software engineering educational system. *International Journal of Engineering Education, 24*(4), 681-688.
Yung, E. (2013). AURALBOOK (Version 2.0) [Software]. Hong Kong, China: Playnote.

Appendix A

Questionnaire: Implementation and Expectations of Music Software from Music Teachers' Perspective

This questionnaire is part of a research project that investigate the most useful and desired features of commonly used commercial music software from the perspective of music teachers. We would appreciate if you can spend about 3 minutes to fill in this questionnaire and participate in this study.

1. Gender: □ Male □ Female

2. Education Level: □ Doctor □ Master □ Bachelor □ Diploma

3. You are teaching in: □ International School □ Special Needs School
   □ Secondary School □ Primary School

4. Which type(s) of music software have you used to facilitate your teaching?
   □ Music notation/scoring software (e.g. Sibelius, Finale, Overture)
   □ Music sequencing and recording software (e.g. Logic Pro/Express, Pro Tools)
   □ Audio editing and recording software (e.g. Audacity, Sound Forge)
   □ Theory & aural training software (e.g. Musition, Auralia)
   □ Audio loop composing (e.g. Garage Band, Acid Pro)
   □ Step sequencing (Reason, Fruity Loops)
   □ Auto arranging and accompaniment software (e.g. Band In A Box)
   □ Elementary music teaching software (e.g. Music Maestro, Jelly Beans)
   □ iPhone/iPad/android apps
   □ Others:
   _____

5. What are the sources of acquisition of knowledge about music software?
   □ University course work/study □ Music/software retailers
   □ Internet □ Colleagues □ Friends
   □ Mass media (e.g. newspaper, music magazine) □ Others: _____

6. How do you rate your computer proficiency? (5 being the best)
   □ 1 □ 2 □ 3 □ 4 □ 5

7. How would you rate your confidence in using music software for teaching purposes?
   □ 1 □ 2 □ 3 □ 4 □ 5

8. Which of following music activities do you think most suitable for the integration of music software technology to facilitate the teaching and learning of music?
   □ Music history □ Music theory □ Composition
   □ Aural training □ Music appreciation □ Instrument/vocal training
   □ Others: _____

Appendix B

Interview Transcript: Mr George Mitcheson

Researcher: What software engineering techniques are taught in the software engineering course?

Mitcheson: It's a broad course, which teaches the full spectrum of software engineering activities. So there are some aspects of project management, some software processes, principle activities, which form the software engineering process, those tasks which form the activities, and the introduction of a number of special techniques in the areas of design, analysis, requirement capture, (software) testing. We have a student project as well, to get some hands on experience.

Researcher: What particular software engineering processes did you teach in the course?

Mitcheson: We introduce a range of software processes from the classic waterfall model, the unified process which is the current approach of the software engineering process, and other processes such as the very short iteration processes, such as the extreme models.

Researcher: How would you select which particular software engineering components to teach?

Mitcheson: One thing we can do is to introduce vocabularies. To be a software engineer one hsd to understand the terminology in the field. So we hang some of the material on those terms. You must have some understanding of requirement engineering, and you have to know what that involves. So we pick some items and teach around that topic. You need to know some analysis processes of software development. We have to teach the modeling approach, and we tend to select the new approaches that are not quite the forefront in the field, but are those 'fashioned' approaches – they may not be fully adopted, for example, in Hong Kong, but they are becoming the state of the art in the software engineering companies.

Researcher: What were the sources for updating the course content?

Mitcheson: The core of the software engineering history of course remains stable, we have to give students the background of foundation so they know where the field of development from. And the underlying activities in name haven't change, the structure always contain the requirement engineering,

analysis, design, implementation, testing, etc. To incorporate new materials, we just hang those new trends into those topics and activities. For example, a decade ago design patterns became popular with the publication of the famous design pattern book, so we started to introduce the design patterns with some explanation. More recently, in fact, a new approach was published which built on the idea of the design pattern but took it on to a more environmental level, showing the elements used to construct the design patterns themselves. So I have incorporated this idea. But clearly we cannot discuss the entire topic. So we just introduce this idea with a few examples and give students 'pointers' so that they can know more themselves if they want to. Each year we introduce what have changed in the field, but hang it on the hooks that already in place.

Researcher: What is the relevance of fields such as business and education to the course?

Mitcheson: Very little to be honest. We could teach the business of software engineering, but have some hard choices to make, since we only have 40 hours – actually just 36 hours class time – and so we focus on the more technical side. Yes, we have project management – but project management related to software development, not general project management. So it's managing software processes rather than management per se.

Researcher: Do we have other electives of software engineering nature other than the course Introduction to Software Engineering?

Mitcheson: We have related software engineering electives which grow out from the software engineering programme. For example this semester I am teaching a course Implement, Testing and Maintenance of Software Systems. We have a project management course too.

Researcher: What's the student software engineering project about in the software engineering course?

Mitcheson: Last 3 years this course has been taught by a different teacher. The project at that time is about requirement capture and modeling.

Researcher: How might students put their software engineering knowledge into practice within the course and within the programme?

Mitcheson: Yes of course. I think it is a requirement. If the FYP is anything about software development, the stages and deliverables are all based on the processes they learnt in the course. For example, project steps and keys in the unified process, and deliverables we expected in each stage of the unified process. So it's directly applied. Similarly with testing, we expect students to test their software and apply standards to their software testing. By the way I am a great friend of documentation, I encourage students to look carefully what documents they produced, questioning whether they are worthwhile. If it does

not help the project then just forget it. There is no need to produce documents just because those titles are listed, you don't have to adopt the template completely. I hate duplication of documentations.

Researcher: How relevant are the course materials to real-world practice in the software industry?

Mitcheson: I mentioned that we teach them vocabularies before. In the work place, they will be attending meetings, they have to know what people are talking about. So immediately they're using knowledge that they learnt. They make sense of what they are hearing and what they are asked to do. We find our graduates can enter and start being productive in a new job after graduation very quickly because they have knowledge about what's happening there, what they are asked to test, what that means, what documents they have to produce, what resources they need to do the testing. They are well equipped to start working immediately. What I don't see in Hong Kong is the importance they place in software engineering, anf software development. Although many people will say yes, this is important, when it comes down to allocating time and budget for it, you found that most managers and companies resist. They are not convinced with the usefulness. It is being used but not being used as well as it could. For example, if I have a 5-man software development company, I found most those companies are just rushing to their deadlines, and looking for new projects, and just trying to make a success of their business without focusing on the quality of the products they are producing. It is an issue of quality actually. You know software engineering in principle is to help you to build software with quality. And there is not much attention placed on the quality of development. For small projects such as developing mobile apps, the attention and knowledge of the developers are even less. I mean the process. But obviously the skills that we teach, things like design, analysis, testing are used in app development as well as large scale software development. I think students graduated from the programme with two things. They have knowledge about the process, and skills, knowledge of techniques. They use them in any software development enterprise they entered, but the formal application process and quality insurance might be lacking.

Researcher: What are the differences between what is being taught and how they apply the software engineering knowledge after graduation?

Mitcheson: Yes, I can give a couple of explicit examples, may be not in the Introduction of Software Engineering course, but in some of the other courses I teach. In the master level programme, I encourage test-driven development, or in refracting I encourage unit test in place. So we write the tests as well as developing the specifications, and we write software to pass the tests. You can't do things like refracting without testing framework in place and unit tests written that you run it again and again when you change the software. So this is acknowledged to be the way to develop software. So I teach them and I have students practice test-driven development and developing unit tests based on

testing frameworks such as Junit – which is pretty the industry standard. Outside there is no time to write the tests. Everyone knows this is a good idea, but your manager won't give you time to write those unit tests. So you have to do it yourself, if you are skilled or trusted enough to do it yourself – you make your own individual decisions or you want to make a contribution to make a high quality software you will back it up with unit tests and use a unit test framework. But you are not forced to do that by your company, and probably your boss discourage you because if you are doing that you really have to do something else. It means you have too much time and you should be using that to write more codes. Another aspect what I see is in requirement engineering. Everyone in the software development field knows that most of the problems they face aren't technical. They come from misunderstanding requirements or poor requirement engineering. And yet still even though I took a large amount of time of the course to discuss requirement engineering, how to capture requirements, how to document them, how to model based on them. In the industry there is not much attention placed on that aspect. You go in quickly, you think you understand the problem, you go away and build something that you think can solve the problem, and then you fix it when it's wrong. That's still the attitude of developers in Hong Kong.

Researcher: How relevant is software engineering education to music software development?

There are a lot of similarities of what you are talking about and what is in the field of architecture. I think music is a little bit architecture because architecture has been using computerised development. They develop their own technologies, may be 5 or 6 years ago the Walkthrough became commonplace where you can build the 3D environment and actually walk through it virtually. They used those visualised designs. I have students working in architecture and I have seen the technology is changing the way that architects perform. It's a creative pursue, but becoming almost impossible to achieve a large scale without the support of technology – without these tools. So architecture training is incorporating these technologies. Just as you said music education training must incorporate those technologies.

Appendix C

Interview Transcript: Prof Tse Tsun Him

Researcher: What software engineering techniques are taught in the software engineering course?

Tse: Mainly the unified process and unified model language.

Researcher: What particular software engineering processes did you teach in the course?

Tse: The software development cycles being taught are based on the unified process which I do not explicitly teach in undergraduate courses but it is still there in the sense that it is iterative. I concentrate on software analysis, design, and testing.

Researcher: How about other models such as the waterfall model, extreme programming?

Tse: I tell them, but the waterfall model is outdated. I personally feel that extreme programming is so extreme. So in Hong Kong, they don't normally use these kinds of methods. So I just teach things that they can find a job easily.

Researcher: How might students put their software engineering knowledge into practice within the course and within the programme?

Tse: They got joint project. A team of 4 or 5, and then they work on analysis and design. We do not ask them to implement the system, so they just use IBM rational to draw the analysis and design, diagrams and that's it. Because I feel that they already have a lot of programming courses, I don't need to teach them those programming methodologies. It's also because of the time constraint.

Researcher: You have also supervising some of the FYPs. Do they really apply the software engineering knowledge?

Tse: Actually they apply because I ask them. Undergraduate students do not fully appreciate real world practice even though I do my best to convince it to them. Given the chance, they would simply write code. But then once they go outside in the real world, they will find it is really necessary to work on the specifications.

Researcher: So the reason is they do not realise the importance in their FYP/study?

Tse: That's right. But once I ask them, then they follow. I guess it is similar when they work outside, because the boss requires them to write the specifications, and then they write the specifications.

Researcher: How relevant are the course materials to real-world practice in the software industry?

For example, every year some students ask us, do the industry really use IBM rational? Because they are too difficult. So we did ask a lot of people like the government people, semi-governmental bodies, large industries, HKIE, etc. And they all say yes.

Researcher: What were the sources for updating the course content?

Tse: From two main sources: From books, and papers, and published UML standards; and from real-life experience through my industrial projects and contacts with former students. For example, quite a number of former students come back to me to say hello. I would ask them about the trends in their software engineering practices.

Researcher: How often is the course being updated? Is the change of the software engineering discipline fast?

Tse: I tend to make minor changes every year. So does industry practice.

Researcher: What difficulties do you find in teaching students software engineering?

Tse: I teach both MSc and BEng students. I found that it's much easier to teach MSc students. Because a lot of them have been working in the real world industry. They understand what the difficulties are. And then when I point out the difficulties, they understand and trust my teaching, and work accordingly. But then for undergraduate students, since they have never worked before, they can do programming assignment easily but they do not understand why they need to do so much in specifications. So they have a certain kind of distance from the course, thinking that what we are teaching is only academic. One example. I have a student who was in my undergraduate course and he did not like software engineering at all. He started up his own company, because I told him large companies would all have those UML standards. But then many years later he came back to me, saying that 'Now I am the expert in UML;, because he's having a small company, so he has to be the sub-contractor for large companies. So every time he has to work on all those UML standard specifications and submit to the large companies and have their approval. So now he actually becomes an expert in UML. Every year I try my best to have some experienced professionals to come to HKU. Because they are my former students so it's quite easy for me to invite them to come back and then present. And once students hear their version of how those specifications work, then they trust my version. Otherwise they will think that I am too academic.

Researcher: So the difference between master students and undergraduate students actually is the real world experience, and that's why the real world experience matters their perspective towards software engineering?

Tse: That's right.

Researcher: There are some FYPs dealing with developing apps for smart phones. Do those projects also make use of software engineering?

Tse: If they are working under me then yes, but if they are working under some other supervisors who themselves are not aware of UML or those kind of standards, then it may not be the case. So it depends on the supervisor.

Researcher: At the FYP level, do you see the differences (between making sure of software engineering and not)?

Tse: As far as the software product is concerned, the difference may not be that obvious. One of the advantages of using those documentations is once you put them in the production stage, when user asks for enhancement, maintenance and so on, if you have the documentation then it will be much easier. For the FYP, once it's done, it's done. So there is no question about reuse or related issues. So if one wants to do a big project, then software engineering is important. Yes. When it's a small project, one or two persons' project, then any method will be good enough.

Researcher: In my literature review, there are many innovative practices trying to get software engineering students outside to the industry, or use other methods such as problem based learning, so do you consider those approaches and do you have any case study in your course?

Tse: Actually we do have some of these internship and FYPs which are connected with the industry. So these students will work on real life projects and then they would apply our software engineering technique to those projects. My FYPs are more research-oriented where students will think about the problem and how to solve the problem for research. I would think students should be trained in both aspects academic and practice.

Researcher: How do you assess students' performance on the software engineering course?

Tse: 50-50. 50% based on projects and assignments, and 50% based on examination. They will do one analysis and one design for the project. Actually what we do is, one week before the deadline, we tell them 'we as users have changed our minds, please follow the changes'. It's real life.

Researcher: How relevant is software engineering education to music software development?

What I can see is those software developers and music educators do not really communicate with each other. And there are so many functions that the software developers developed are of no use to those educators. They just use the basic functions. That's something that I find it especially difficult to teach for undergraduates. Because I ask them to interview the users who are the tutors of this course, and then after the tutors have given them some requirements, the students like to make up a lot of additional requirements which are not required by the users. The other problem is those developers are trying to get everything from the hardware to the software. May be you can have a software mixer instead of a hardware one which could cost less because everyone has a computer, but it actually do not help because it's the same thing. So the developers have something in their mind they want to sell to the music educators, but they should first of all ask the educators what they want

first. Those original ideas are actually come from those software developers, they do not come from those music educators. I know there are some difficulties for them to tell what they want. We do not need to ask music educators about what the computer-related requirements are. We should ask them for the music-related requirements. And then the analysts and designers translate those music requirements into deliverables of the software. They should not have a preconception of what music should be like. For example, they should not have an assumption that music educators should use iPad. So the music educators should have something in their mind and then software designer will design accordingly to the requirements of music educators. For example, for the latest design they would think of iPhone and iPad, but the music educators may be thinking about something without keyboards. Then developers should think of something without the use of keyboard. If it is not feasible then they do something else. But first of all they should listen to what the music educators want. They may like to have something more conventional – it looks like a piano or a trombone rather than something which is more look like a computer.

Researcher: The problem there I think is the communication protocol between stakeholders.

Tse: So the best thing is the music educators don't try to think like computer. They would just give their music opinion. And it is the software people's responsibility to convert those things to computer things. And it may not look like the current computers.

Appendix D

Interview Transcript: Mr Eric Yung

Researcher: What are the goals and visions of your product?

Yung: To help the teachers to teach aural skills in a better way. Because in the current teaching environment, teachers do not have time to teach aural skills in musical instrument lessons, and most of the time is spent on teaching instruments. The aural skills are somehow difficult for some of the teachers who do not have the formal and complete music training. For example, for those who do not have a degree in music, but just have a practical certificate or diploma, the skills that they learnt were concentrated on their instrument skills, but not about musicianship, music history aural skills or whatever. So they have difficulty in aural skills, no matter in singing or recognising some music or talking about music history. So this software will help the teachers to teach in a better way, both benefit the teachers, students and parents.

Researcher: Is your software teacher-centred or student-centred?

Yung: Because the teachers do not have time in the lessons and they cannot teach in a good way. So what the teachers do is to introduce the software to the students and ask students have the self-learning at home, and the teachers can just give some introductions in the lesson, and the students can do it by themselves.

Researcher: Who are the targeted users of your software?

Yung: Students.

Researcher: What are the competitive advantages of your software?

Yung: First of all, how to define aural skills apps? I know there are lots of ear test-sets. In an ear test-set you listen to something, and then you decide if it is a C major chord, or a crotchet or a minim. That's it. Can this be classified as an aural skills app? From my point of view, it is not. It can be called a grade one music theory app because it teaches you basic skills of music. It is still teaching, but not about aural skills, which are about singing, clapping and listening, with lots of music knowledge, music history etc. So up to now, I haven't seen any app in the Appstore that is directly competing with us.

Researcher: Have you considered releasing your product in Android?

Yung: Should be within January – it is in the final testing.

Researcher: Through what channels do you sell your product? How do you advertise your software?

Yung: Marketing is somehow the traditional one, that is, no matter what is the product, the marketing is nearly the same – you need to send your message to the real customer in the most efficient way – that's marketing, right? So what I can say is that online – which means Facebook, email or any online marketing campaign, or offline – advertisement, sponsorship of some concerts, that's it. Those are not related to the product because no matter what product it is.

Researcher: Do you have any corporations with schools?

Yung: Not yet in this stage, because our product is not as complete as we think – because as you have said, there is only iOS version. Unless we need to complete the whole syllabus, the majority platforms that we will work with, then we will see how we can collaborate with those schools.

Researcher: How do you provide user support to your customers and how do you gather feedback from software users?

Yung: There is a lot. The majority is by email. Because you can say telephone, whatsapp, Facebook, etc, email is still the major communication tool for normal users. So we can receive many emails a day, and of course different kinds of questions, we just reply them one by one.

Researcher: What problems have users encountered when using the software?

Most of them we just use standard replies – because the problem, or what we called the difficulty of the majority of users are the same. For example, they will say that they cannot hear the music. We just copy and paste, try to turn on the speaker, something like that. Another example, I cannot connect to the server, okay please check your Internet access and whether the wifi is turned on. Most of them are technical ones, because unlike other products which are for tech-lover, our products are for students, teachers and music-lovers. Music-lovers do not mean they are tech-lovers. So they just buy the iPad as a PC, but not iPad. They downloaded the app, but this is for tech-lovers, tech-lovers can solve this kind of problems but music guys may not be able to solve it.

Researcher: What motivated you to develop the software?

Yung: That sounds like the very first question. This is because I saw the problem in the music education industry market, and I know the technology is ready now, so there is a demand, so we provide it.

Researcher: So it started with your observation of such opportunity?

Yung: Because I also teach music by myself, actually I know this problem from a long time. Even I am a student, I learnt piano, the problem also existed even 20 year ago. But just no solution. And so I just wait and now, the technology is ready, so I started the product.

Researcher: What software engineering techniques did you use in developing your software? Did you conduct any requirement analysis?

Yung: That's somehow tricky. Because, first of all, our software must fulfill the examination syllabus of the authorities, that's the guideline already. It sets out all the steps, even what they say, how they ask, it is all set. Somehow this question is not truly related to this specific product.

Researcher: Did you conduct any software testing?

Yung: Yes, but somehow it is difficult. Just like Apple philosophy. The users do not know what they want. But when they see the product, they know what they want. The same here. No one has ever think what they want to try before, so actually no one knows what is the requirement. So the only fact is that, when it is completed, people know, okay, this is what we needed. Our approach to the user requirement is somehow trial and error. There is no formula or formation of getting users requirement because the user cannot tell you. We just by do it iteratively, make it, and then test it.

Researcher: What software engineering training have you received?

Yung: I can say so because I graduated from the electronic engineering, both bachelor and master, and so you can see this as a foundation of training. But somehow the real training is in the commercial world, not academic. We forget any software development models, we just develop the software by ourselves, any existing model does not fit into our product that cannot be categorised. What you are saying is about – okay we have an ERP software, CRM software, there are some innovations in it, or better features compared to the existing product. But the core is the same. There are some workflow or version control or deployment system. Then you draft the specification, verify it, the system analyst does it, the programmer make it, okay that is the process. But I can tell you that in the development of this software, there is no product specification at all. We just write code by code, trial and error, because something you can think about but cannot be implemented. So you forget it. During the development, you find out something that can be developed but you have not forecasted before so we add it immediately. If there is a product specification in it, the level of innovation will be much lower. Because it is restricted by the system, and every engineer will just try to follow the rules, not to do a wrong thing, and then it is the standard product.

Researcher: What were the biggest challenges in developing your software?

Yung: It takes one and a half year for the first release version. Biggest challenge is the iOS is a closed system. Everything is controlled by Apple. So you need to use a lot of time to negotiate with Apple. You need to understand what Apple is making, and the language that they are using.

Researcher: So you have the same challenge with Android?

Yung: If you compare Apple and Android, I still prefer Apple. Even though the business of Apple gives us a lot of troubles, the technical development, the experience is better with Apple. Because Android will tell you that you can implement anything, we will not restrict, but somehow it is difficult to implement because of its flexibility.

In Apple it is restricted, so actually other than your core technology, for other things no matter user interface or hardware it is restricted. You have no other way to go out, if you just follow their rules, then that's fine, it is easy. For example, in Android, just compared with Apple, here is one resolution, here is another resolution. Samsung Galaxy has one screen size, then your implementation shall ensure that the graphics need to fit in every size. And then Android machines have different sizes of memory, different speakers, you need to ensure your software can adopt different combination of hardware. From our record, the current Android database has 1800+ devices registered in Google. So are you go to adopt 100? 1000? or all? Or even include those unlisted devices manufactured in China?

Researcher: What constitutes good music software development?

Yung: I will tell you that the team is the most important. Because the hardware is on the market, any system or software can be changed at any time, but the attitude of the team members and their knowledge is the most important. Because yes, you can have a lot of systems and a lot of hardware to restrict the engineer to do anything and they must follow you rule, but if the attitude is not good or the team does not have enough knowledge, they cannot implement any good software. You cannot monitor any code the engineer is writing. You only see the result. Then if the result is wrong, you just ask 'why' and the engineer can just say 'okay I have implemented it I don't know why', then he trace the code or you trace the code?

Researcher: Can I say everyone in your company knows all the codings and algorithms of the software?

Yung: No. Different team member has different expertise, however, there is no one dominate the others. We just tell them "okay this is the way to connect the module to another one, just do it". If we set too many rules, actually the outcome will not be good.

Researcher: You have a number of patents which related to music and technology. Do you use that in your software?

Yung: In commercial world, paid patent is important. Unlike in academic, the final goal is to find some new thing, to give the impact to the world, and the academic world recognise it, then that's the end of the story. And then you start another round. But in commercial, you still need to think 'can you pay the salary next month?' or 'can it generate enough revenue?' We are not aiming to sue anyone outside, but at least we need to protect ourselves.

Appendix E

Interview Transcript: Dr Ann Blombach

Researcher: What are the goals and vision of your software?

Blombach: It's mainly ear training, but I've added basic practice in intervals scales and chords – written and keyboard. It's to provide students with something they can do on their own to help themselves learn these skills.

Researcher: so could I say it is aural training software?

Blombach: Mainly aural training, and I think that's how most people think of it. The 'theory' skills are all things that could help the student with ear training as I think of ear training. Helps them learn how to spell chords, etc., and the keyboard skills help their ears as well.

Researcher: Who are the targeted users of your software? Is your software teacher-centred or student-centred?

Blombach: It's designed for students' self-learning. Something they can work on entirely on their own. Teachers can 'direct' it, though, if they want to, because they can change a lot of how MacGAMUT operates. It's been used by grade school students as well as graduate students, and that's because of the way teachers can alter how it operates. I should say it's used mostly in high schools and colleges/universities.

Researcher: What are the competitive advantages of your software? There are so many ear training software in the market, how would your product 'win' the rest?

Blombach: I really care about teaching and student learning, so for me it's more than just something I sell. I'd give it away if I could afford to do that! I think the biggest advantage is the intelligence of the grading of melodic, rhythmic, and harmonic dictation, along with the musicality of the exercises. I think the fact that it's mastery-based is also important--make sure the student can do the work for each level before going on to the next. And finally, the fact that I taught theory and ear-training for 25 years at Ohio State, and tested it with my students for many of those years. I know it works.

Researcher: Through what channels do you sell your software product – other than from the official website?

Blombach: We mainly sell it to bookstores, just like textbooks are sold. An instructor adopts it for his/her class and then orders it through the bookstore. We figure our real cusomers are the instructors, they're the ones we really have to convince. We have some individual sales as well, but it's mainly because instructors tell their students to buy it.

Researcher: How do you advertise your software?

Blombach: We like snail-mail, primarily because most people ignore email these days. We have about 3,000 instructors on our mail list - people who are using it, but also people who have just expressed an interest so far. When we have something new, we do a big mailing to the College Music Society list of theory and aural training teachers in colleges and universities. And we also mail to high school teachers who teach music. Finally, we have instructors who believe in MacGAMUT who do a lot of Advanced Placement workshops. At this point, we also have a lot of people who used MacGAMUT as students, and they're now requiring their students to use it.

Researcher: How do you gather feedback from software users and provide user support to your customers?

Blombach: I always say that that's why MacGAMUT is as good as it is. I take comments and questions very seriously, even if my first reaction is that something the person is asking for is just crazy. Like any other teacher, I used to think my way of doing things was the only best way. But the more I worked with students and other instructors, the more I realised we don't really know how people learn aural-training skills, their way might be just as good as mine. That's why MacGAMUT has so many options and is so flexible. From time to time, I also work closely with individual students, not just instructors, because a student will suggest something good, and then if I encourage that student, he/she keeps suggesting things. MacGAMUT is all mine, but the ideas are most definitely not all mine. There have been so many comments and questions over the years. A lot of the security features are a result of my watching my own students work in the lab. I could see how they were trying to get around this or that. That's why they can't quit before they have their answer checked at least once in mastery mode. Rhythmic dictation was something instructors wanted badly, along with beaming. Right now, I'm working on setting up a quiz mode as well as straight mastery mode - again, something instructors wanted.

Researcher: What problems have users encountered when using the software?

Blombach: I try really hard to make sure there aren't problems before I put anything up on the web or on a CD. And if someone reports a problem, I am almost always much more concerned about it than they are. I Hate it when anyone has any problems. I've heard of other software developers who don't even take bug reports seriously when they're beta testing, let alone when the real thing is out there. That will never happen with MacGAMUT. I know there's no such thing as perfect software, but I do my very, very best to come as close as possible. We have probably 25,000 users out there right now, and our tech support staff consists of 1 person who gets maybe 3 or 4 tech questions a day on the very worst day. And it's usually someone who doesn't know how to download a file from the Web or doesn't know how to put a CD into a CD drive. Almost never something to do with the software itself. I just realised, I can actually tell you how many real bug reports we've had this year so far - a total of 3, and one of those was because an instructor wanted to do something I'd never thought of doing. So more like an enhancement request than a bug report. I think it helps that I'm the only programmer and have always been the only programmer. If something goes wrong, I know it's my fault. I can't blame it on anyone else. So I have to fix it!

Researcher: What software engineering training have you received?

Blombach: Actually, no. I took computer programming in college and did not do well. I was a math major – undergraduate degree is in mathematics – so it seemed like something I should do. But it was on punch cards, and when you turned your program in, it took a week to get the results back. I tended to get into endless loops, and since it took a week to find out, I was in big trouble. So my first success at programming was on the Macintosh since I could find out right away whether I'd done something wrong or not. And that I could deal with because I wouldn't give up until I got it right. I did do some independent study in computer programming as a grad student, but except for that terrible first course, I've never had another actual programming course, although I've taught them many times. I'm very much self-taught.

Researcher: Then do you have ideas about software engineering? What were the biggest challenges in developing your software?

Blombach: I'm unusual in a lot of ways! A lot of the music students who took my programming courses went on get computer programming jobs in the real world, and most of them were very happy and successful, so I think I must have done a good job of teaching them. I think my main opinion is that software engineers should test their software more thoroughly! Just because it works for them doesn't mean it will work for everyone else. They should also test their software on every kind of machine and system they can possibly test it on. I know team programming is the way most people do it, and I think that's generally a good thing because one person doesn't have to know everything. On the other hand, though, I guess I'm a control freak, but I really want to know every little detail. I don't see how you can be sure the software will work if someone in charge doesn't know exactly what each person is doing. But I'd rather do it all myself than be a team leader. More than you asked, right?

Researcher: What software engineering techniques did you use in developing your software?

Blombach: I try to lay out very clearly what I want to do before I start programming because otherwise, it wastes a whole lot of my time. But that's the closest I get.

Researcher: What constitutes good music software development?

Blombach: I'm probably not a good one to answer that question since I do my own thing and answer to no one except myself. I tend to work as much as is humanly possible. I've learned the hard way that the last 10% of development takes about 90% or more of the time, so I try to keep that in mind when I'm deciding how much I can get done in time for the release we have planned. Of course, then I need to allow a lot more time for testing after that. I'm not a 'team' person in general, although for the last year and a half, I've been working closely with another programmer on an eText we have just published – the first non-MacGAMUT thing we've ever published. The two of us work extremely well together, so I can no longer say that I only work alone. Maybe I just hadn't found the right programming partner before. We have to make our software available for both Mac and Windows, so I develop on both platforms. As for the environment, I work in XCode for Mac, Visual Studio for Windows – the

accepted environments, in other words. In the collaborative project with Elizabeth Sayrs, we work in Director, Flash, Finale, Audacity, and other programs. She works on Mac, I work on Windows. Except for font issues, the only time the platform matters is when it's time to put the icon on the application – that has to be done on the platform it will be used on.

Researcher: To you, what are the most important factors to consider when developing your software?

Blombach: What students and instructors need is always important. Also extremely important to have software that is as trouble-free as possible. I think those are probably the two very most important factors. On second thought, I have to admit my own bias. Having taught theory and aural training all those years, I have some definite ideas about how I think software should be used. It should never be used as a substitute for a teacher. From time to time, a school decides that if there's software, then they can just do it all online or on a computer, and they can fire an instructor. That's insane. Even the best software can't substitute for a human being who is a great teacher. All it can do is free up classroom time so the teacher can do what only the teacher can do. Not what you asked exactly, but very important to me that there never be any confusion about that!

Appendix F

Interview Transcript: Dr William Thorpe

Researcher: What are the goals and vision of your software?

Thorpe: Sing & See is to improve teachers' and singers' vocal training experience. It has a vision to research the effectiveness and efficiency of incorporating real-time visual feedback to singers.

Researcher: What are the competitive advantages of your product?

Thorpe: I think the main thing is it has a very actual pitch recognition algorithm, and it gives pretty good feedback to the singing voice. Also the way we develop – we develop it with singing teachers, so it was designed to be easy to use, which give back to your point you made before that some software hasn't have that design interaction with actual users so it is not so easy to use.

Researcher: Who are the targeted users of your software?

Researcher: Singing teachers and students.

Researcher: Through what channels do you sell your software product? I know I can buy Sing & See from the website, is there any way that I can purchase Sing & See other than the website?

Thorpe: No. We had had some collaboration with retailers but it hasn't really work out, probably because it's an inch product. So online is the easiest way to sell.

Researcher: How do you advertise your software?

Thorpe: We do online marketing, and we also do some advertising in singing education journals.

Researcher: Do you release your product in a virtual way through the Internet instead of sending physical copies?

Thorpe: Yes, it can be bought or download as well.

Researcher: How do you provide user support to your customers and gather feedback from software users?

Thorpe: It's mainly through email and we also use fogbugz. It's an online issue tracking system. And we also use copilot which makes me interact with the users' computer that I can work through on their computers. And we set up some online

forum on the website for help as well. But it hasn't been used so much compared to email.

Researcher: What problems have users encountered when using the software?

Thorpe: The main issue they come out is issues with audio, because that's kind of the problems that might stop them to use. They might not be able to see the microphone, they might not see the audio coming in. It's usually the issues of configuration with audio.

Researcher: So the problems are mainly technical problems, right?

Thorpe: Yes.

Researcher: What kind of information does the product decision based on? Did you conduct any requirement analysis or software testing?

Thorpe: We do user surveys, so we have feedback from users. Also, the comments that people volunteer, and interviewing and talking with users one to one. So it's feedback from users generally.

Researcher: What motivated you to develop the software?

Thorpe: It is something that I am always interested in, and I was actually doing research in this area in the university, and set up a project looking at visual feedback software and singing training. And had a lot of positive feedback from teachers and singers about the prototype we developed, so I used that as the basis for the product.

Researcher: So it started with prototype?

Thorpe: Yes. It came out as a research in the university.

Researcher: What software engineering training have you received?

Thorpe: My degree is in electrical engineering. It wasn't any software engineering. So I guess I have been a computer software development on my career, but it hasn't really been called software engineering.

Researcher: So you learn programming from your bachelor programme?

Thorpe: Yes, we had programming. It was Fortran and Pascal.

Researcher: What software engineering techniques did you use in developing your software?

Thorpe: We used an iterative approach – developing the prototype and then getting feedback and then refining it. It's not that traditional design driven approach.

Researcher: So many documents come out at the beginning for the prototype before you actually code the software, right?

Thorpe: Yes, we have a rough skeleton of how the programme goes and then improve it.

Researcher: What constitutes good music software development?

Thorpe: One of the things I notice is the interactions with singing teachers and music teachers. Their priority of focus is on students and on the music. So any software tool they use has to fit in with the actual music, and it's got to be very simple to use – they want to focus on the students and the music rather than on the counter. And there are constricts in the actual teaching situation. In the practice situation you have a singer or student who are practicing, the focus is on the music, the software can give some feedback but it's got to be done in a natural way that it doesn't take away too much concentration from the learning.

Researcher: Too much information given to the singing students will actually decrease their performance.

Thorpe: Yes. So you have to give information in a relevant way.

Researcher: Is your software teacher-centred or student-centred?

Thorpe: Yes, the students got to interact with it, adds to the learning experience instead of getting out from it.

Researcher: What constitutes good music software development?

Thorpe: You got a good team obviously is a very important factor. And also the design, if you do not have good software interface, this impact the ability of users to make use of the software. The software should also have a good architecture so it's easy to improve and upgrade.

Researcher: Do schools use Sing & See in the classroom context?

Thorpe: We really have not yet focus on the school situation, because the software is much more about individual feedback, so it is kind of a one-on-one product. So it hasn't really fit very well into the school situation where the teachers teaching class or you got a class or people playing music. We got a few schools and universities starting to use it in the class situation, but they already got individual computers. But the majority of music classrooms don't have it for music, or they might have it for music theory but you can't have the whole class singing (to Sing & See) because it will interfere with each other.

Researcher: Is that the students use it more for self-practice?

Thorpe: Yes.

Appendix G

Interview Transcript: Mr Kenny Lui

Researcher: What discount do you offer to music teachers and students when purchasing music software?

Lui: Yes. Many music software suppliers provide educational prices for teachers and students. Students can purchase the educational version of the music software by showing their student cards.

Researcher: Is there any educational version of the music software that Tsang Fook is selling? What is the difference compared to the regular commercial versions?

Lui: Most of them do not have any differences. Their functions are the same. They are just discounted for eligible schools, teachers and students.

Researcher: What are the occupations of the participants of the music software workshops? Are they mainly teachers or students?

Lui: Our customers range from knowing nothing about music technology to music professionals. Their interests range from using music software for fun to earning a living.

Researcher: What questions do those customers ask when they purchase the software? What are the customers' key concerns and criteria when purchasing music software?

Lui: First of all, they consider the purpose of their software usages. Then they will consider the ease of use compared with other music software that serves the same purpose. They will also search the software reviews on the Internet, or download the demo to try and compare.

Researcher: What problems do those customers raise during they purchase the software?

Lui: They would question whether the software is capable for their purpose. If they want more functions, they would buy the full version. If they think the light versions are enough, they will buy entry level software. Then they will ask whether the software is user-friendly or not.

Researcher: What problems do those customers have after they have purchased the software?

Lui: Most problems they have after the purchase are technical problems. They don't know how to install the software, there are system errors after installation, or they don't know how to connect the software to hardware. We do have technical staff to

answer those questions. But if they want to know how to use the software, they need to attend courses. We provide night time courses, or users can seek for courses outside. If there are problems that we cannot immediately solve, or there are problems that we cannot answer from the perspective of retailer such as registration problem, then we will help them to ask the software developers or refer them to the developers.

Researcher: How do you receive feedback from customers?

Lui: Mostly by emails and phone calls.

Researcher: Within the same functional category of music software, there are many choices. What factors affect the company's choice of music software developer/supplier/products?

Lui: The acceptability and popularity of the software, whether the software is the leading brand among the market, and the potential of the software in terms of future sales.

Researcher: How do you provide feedback to software developers?

Lui: Customers sometimes provide feedback about potential improvements of the software. Mostly we do it through email, and sometimes at tradeshows – LimeZone in the US, or other tradeshows in Frankfurt, Beijing.

Researcher: What are the purposes of going to those tradeshows?

Lui: Mostly for updates from software developers. We will talk about the software sales, marketing and sales strategies later.

Researcher: What kinds of information do you provide to music software developers concerning their software development process?

Lui: Details about the local market, e.g. which products are popular in the local market. Citizens from different regions may have different preference of software usage.

Researcher: From the perspective of music software retailer, why would Sibelius be more successful than Finale?

Lui: The main reason is the user-friendliness of Sibelius over Finale. We didn't really push any particular product nor apply any marketing strategy on particular products, at the end it is the quality of the software product that matters.

Researcher: Who are the major competitors in the music software business?

Lui: Tom Lee. We do not have many competitors in the local market. Actually, our largest competitor is the online market. The Hong Kong local market is small and as the Internet grows more popular, customers may switch to the online market to search for music software. We always monitor the software prices from online stores to make sure our selling prices are competitive enough.

Researcher: What are the recent trends in the music software business?

Lui: More and more software developers are offering download version of their software products. The box versions are getting smaller and smaller, the user manual is starting to disappear in those software packages, since customers can download them from the Internet. If the file size of the software is not too large, we would gradually offer download versions on behalf of the software developers. The main advantage of offering download version is the shortened trade duration. We do not always have box version available in our shops. Once there they out of stock, we have to reorder from the software developer which takes a lot of time. The duration that we get the download version from software developers is a lot shorter than box version, because of the logistic problem. Download version is just all about serial number. This can be done through email. Moreover, download version is normally cheaper than box version because of the decrease in transportation cost. The trend is that we will sell more and more download version, and the acceptability of download version from the customers has becoming higher recently. Before that customers would feel uncomfortable if there is no physical product on their hands. Now just very few customers still have this concept, and the majority has acceptable 'virtual' products already. The most important factor now is the speed of trading. Customers are not willing to wait now and would like to use the software product immediately after paying the money. This is what we experienced in the past year. This is actually a good change, because we now are more available to offer instant delivery to customers. And we can stock less box version to save physical spaces and avoid outdated software not being sold.

Researcher: What about Apple, Inc. who offers Logic Pro and Garage Band which are quite popular in the music software market? They only sell those software products in authorised stores.

Lui: Yes I would agree. Undoubtedly Logic is recently a very popular software for arranging and composing music. And the majority of our customers – musicians – mainly use Apple computers. The company's policy limits the developers and software providers to sell their software in authorised stores. We can only compete by supplying other products of similar functions. We are not big enough to beat with an international company such as Apple.

Researcher: What training and workshops does your business provide?

Lui: We mainly target teachers. As schools always buy a volume license for software – maybe 40 at a time, or half a computer room, 20. Mostly music teachers and IT teachers. IT teachers usually support music teachers on technical issues, that's why we also target IT teachers.

Researcher: Do the participants know about the music software before they come to workshops?

Lui: Most participants know the software before they come. They come to know more.

Researcher: What problems and questions do participants usually have in the workshops?

Lui: The compatibility of the software with other software and hardware – for example from Sibelius files to Finale files. Questions do not mostly raised during the Q&A session of the workshop, but some questions are being raised after the workshop.

Researcher: Many of the company's music software workshops target at music teachers and educators. Why?

Lui: Compared to the retail market, the education market is much larger, particularly software such as music notation software. The second reason is that only schools are eligible to purchase volume licenses. Groups of random customers are not authorised to purchase volume licenses. A school license has restrictions. We do pay a lot of attention to the education market. The market private or professional user market is too small, especially for holding workshops for them.

Researcher: What features have been made to the workshops to make them more educational?

Lui: Those workshops focus on the applications of the software into educational setting. For example, if we talk about Sibelius, we will focus on features that educators are interested in. We will not talk about the quality of the sound library. That does not interest the educators. We will talk about how to use the software instead.

Researcher: Are the software suppliers interested in or willing to hold/join those music software workshops? If so, what are their roles in those workshops?

Lui: Software developers seldom hold workshops themselves. They will rely on the local distributors, or sizable schools in other countries. They will actively request us to hold workshops or seminars, maybe once a year. Some of them will have cash sponsorship or souvenirs.

Researcher: Software developers rapidly update their software and release new versions of their software products. What is the view of those software updates? Do customers really update themselves to purchase new versions?

Lui: Yes. Some customers do think that the new features provided by the updated versions are useful. But whether those updates worth or not depends on the new features provided. For example, Sibelius does not release new version annually, yet Finale does. And old version of Finale cannot open the score file generated by new version of Finale. Comparatively, Sibelius is more flexible. You can save the score file compatible to old versions of Sibelius. Finale is kind of forcing you to buy new versions by not providing this feature. This is their business strategy. Yet they do have update campaigns – may be 10% update discount.

Researcher: What software do the music teachers buy mostly?

Lui: It depends mainly on what software do they really apply to their teaching. Sibelius and Finale are mostly appeal to music teachers. Other applications such as composing and arranging music, their purchasing behavior mainly depends on which software they used to use.

Researcher: What are the most popular types of music software?

Lui: Two main types – music notation software, for example, Finale and Sibelius, and digital audio workstations, e.g. Pro Tools, Cubase, Logic, Sonar and Digital Performer. Other types include plug-ins, for example, sound libraries. Plug-ins are more appealing to professional users such as music producers.

Researcher: Is it possible to have an integrated software to provide most features that musicians or music teachers usually use which includes notation, recording, sampling, editing?

Lui: It is not impossible, but huge rescores will be required. Logic does have staff lines for notation reference, but you know they are just for reference and is not comparable to dedicated music notation software such as Sibelius and Finale. It is about the focus of the software developer. The question here is that whether it is worth to the software developers to have the integrated software to perform all the tasks. Because of limited resources, you can see many software developers are co-operating with each other to redeem the weakness of each other. For example, software developers that develop plug-ins will have their products compatible to the majority of digital audio workshops available in the market. Every software developer has its own weakness and advantage.

Appendix H

Interview Transcript: Mr Javan Green

Researcher: What software does your shop sell?

Green: We do all the major applications like Cubase, Sonar, Ableton Live. We also sell educational versions of those.

Researcher: What are the occupations of the participants of the music software workshops?

Green: Everyone, from playing music for interest, to students, teachers and music production professionals.

Researcher: Who are the major competitors in the music software business?

Green: This place is far away from other music shops, so the main competitors are actually the online music shops that sell the same things as us.

Researcher: What are the most popular types of music software?

Green: Notation software and digital audio workshops.

Researcher: What discount do you offer to music teachers and students when purchasing music software?

Green: Price Discount. Just the price. They are exactly the same product. It is just a discount for educators, teachers and students.

Researcher: Is there any relationship between your business with music education?

Green: We do quite a lot of cooperation with the Confetti Institute of Creative Technologies, one of the UK's largest audio engineering schools.

Researcher: What training and workshops does your business provide?

Green: We don't. But if someone wants to buy something we can advise them.

Researcher: What questions do those customers ask when they purchase the software? What are the customers' key concerns and criteria when purchasing music software?

Green: The price, what sorts of things they are looking for, and making sure that they get the right product.

Researcher: What problems do those customers have after they have purchased the software?

Green: Usually technical questions, such as problems about installation of software, and how to use particular functions.

Researcher: How do you receive feedback from customers?

Green: Usually by email, or by person.

Researcher: How do you provide feedback to software developers? What kinds of information do you provide to music software developers concerning their software development process?

Green: We speak through the Web, to see how they can help us to sell their products.

Researcher: What are the recent trends in the music software business?

Green: We have been trying to promote our business online now. Probably the new trends are about online music business.

Researcher: How do you consider your role in the system of music education software development?

Green: We are one of the shops who sell music software for education.

Appendix I

Interview Transcript: Millie (pseudo name)

Researcher: Which particular brands of music software have you used?

Millie: I use Finale.

Researcher: Have you used other brands such as Sibelius?

Millie: No.

Researcher: So you only use Finale, and that's it. Right?

Millie: Yes.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Millie: I just use it for preparing class materials, like making music sheets.

Researcher: Which software functions have you used for each type of software applied in teaching?

Millie: For teaching, no. For myself, I will use that when I type music score and I would like to listen to the score, so I playback. Or maybe I compose some simple melody and listen to the playback.

Researcher: What additional functions do you wish the software to have to meet your needs?

Millie: I would prefer an automatic translating function that can translate sheet music to an editable computer file. I know there is such technology available, but it is not accurate enough for my needs. I have to change a lot of things to correct the score. I want to have a more accurate conversion function.

Appendix J

Interview Transcript: Victoria (pseudo name)

Researcher: Which particular brands of music software have you used?

Victoria: I use Finale.

Researcher: Have you tried other brands such as Sibelius?

Victoria: No. I use Finale all the way.

Researcher: How about mobile apps? What kind of apps do you use?

Victoria: I use virtual piano to hear the melody on the music sheets or books, to know how it sounds.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Victoria: Prepare class materials, so I use Finale to make work sheets.

Researcher: Which software functions have you used for each type of software applied in teaching?

Victoria: Not much, I just type the score and print it out. That's all for a primary school teacher.

Researcher: What additional functions do you wish the software to have to meet your needs?

Victoria: When I use Finale to make sheet music, the layout is not what I want it to be. I want to generate one page of sheet music so I can distribute it easily to my class. But Finale always manages the score layout based on aesthetics rather than number of pages, generating one and a half pages of sheet music.

Appendix K

Interview Transcript: Cindy (pseudo name)

Researcher: Which particular brands of music software have you used?

Cindy: I use Finale.

Researcher: Have you tried other brands such as Sibelius?

Cindy: No, I only use Finale.

Researcher: How is the software being used? Do you use the music software during music lessons?

Cindy: No, I use it to prepare class materials such as examination sheets. School does not provide any music software, so I can only use it at home in my own computer.

Researcher: Do you use functions other than score typing?

Cindy: No. I only teach primary school students, score typing is what I need.

Researcher: What additional functions do you wish the software to have to meet your needs?

Cindy: It would be good if there is music software that can suggest chords to me for a given melody.

Appendix L

Interview Transcript: Kobe (pseudo name)

Researcher: Which particular brands of music software you have used?

Kobe: Music Ace, Music Ace II, Musition, Sibelius, Auralia, and some virtual instrument mobile apps

Researcher: How is the software being used? Do you use the music software during music lessons?

Kobe: I have taught students to use Finale Notepad. It's free and also because the school only purchased Sibelius 7 but not Finale.

Researcher: Why don't you teach Sibelius provided that the school has purchased it?

Kobe: It's because students do not have Sibelius at home, it's not free. I want my students to exercise the software at home, and Finale Notepad can serve this because it is free. Students do not have much time to stay at school and use Sibelius other than the music lessons. But Sibelius is more user-friendly and the sound quality is better.

Researcher: How about other software?

Kobe: I use Music Ace in lower form students music lessons because it's game-based. I use it during the music lesson, and after the lesson students will go to the computer room and play it. Teachers have to contribute some conceptual input such as rhythm and beat counting when teaching with Music Ace, if not the students will only treat it as a game and learn nothing.

Researcher: Do you use the music software to prepare class materials?

Kobe: I use Sibelius to make music scores and playback the music to students. Sometimes the music textbook packages do not provide the single melody extracts on the CD samples, so I have to make them myself. I use Sibelius to demonstrate the timbre of particular instruments, too, such as the snare drum, which is better than describing them verbally.

Researcher: Which other software functions have you used for each type of software applied in teaching?

Kobe: I try to use the worksheet database from Sibelius. Also I use the transpose function to playback the difference between different keys; or change the instrument with the same melody to show how different instruments sound.

Researcher: What additional functions do you wish the software to have to meet your needs?

Kobe: I want music software that can really teach music. It could help if the software can guide the students how to compose, which saves the teachers' time. Teachers then can input their own knowledge to supplement the teaching function. More listening examples would be good too. I would like to see sound libraries for Chinese instruments in music notation software.

Researcher: Have you used audio editing software?

Kobe: No. It doesn't suit primary school students' learning ability.

Appendix M

Interview Transcript: Stella (pseudo name)

Researcher: Which particular brands of music you have used?

Stella: The aural training software I used came with the music teaching book, but I forget what its name is. I use Finale and Overture for score editing, and virtual instruments for mobile apps. I use virtual instrument during the music lesson for reference – for example, taking the starting pitch of choral singing.

Researcher: Finale and Overture, which one you think is better and why?

Stella: To me, Finale is better. Most of the time I use Finale since its functions are more comprehensive and it can do everything I require. Finale provides hotkeys which is more convenient. It also comes with a list of common ensemble instrumentation setting such as string quartet or woodwind quintet that I can choose. Overture provides one advantage – Chinese version of the software is available. But in terms of functionality Finale is a lot better.

Researcher: How is the software being used? Do you use the music software during music lessons?

Stella: No, I just use it to prepare scores for my teaching.

Researcher: Do you use functions other than score typing?

Stella: Not much, I use it because of score typing.

Researcher: What additional functions do you wish the software to have to meet your needs?

Stella: It requires a MIDI keyboard to connect if I want to connect Finale with MIDI devices. If it could be connected with iPad, it would be better because I don't have a MIDI keyboard.

Appendix N

Interview Transcript: Ada (pseudo name)

Researcher: Which particular brands of music you have used?

Ada: I use Finale and Overture, both for score typing; and virtual piano for me to realize how particular melody sounds.

Researcher: Which do you think is better and why?

Ada: I like Finale more because it is more comprehensive, but I can only use it at my home only. At school I use Overture because it is free.

Researcher: How is the software used?

Ada: I use it to prepare class materials. I am not that good at computer to allow me to do anything with it in the class. But I managed to do some score typing to make some music exercise sheets.

Researcher: Which software functions have you used for each type of software applied in teaching?

Ada: Not much, software typing is what I need.

Researcher: What additional functions do you wish the software to have to meet your needs?

Ada: I don't have an idea…may be I call you if I think of something.

Appendix O

Interview Transcript: Joyce (pseudo name)

Researcher: Which particular brands of music software have you used?

Joyce: I use Sibelius.

Researcher: Have you used other brands such as Finale?

Joyce: No.

Researcher: How about audio editing software and audio looping composing software?

Joyce: Audacity and GarageBand.

Researcher: How about mobile apps?

Joyce: Virtual instruments.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Joyce: Yes for both. Most of the time I use music software to prepare class material. During the lesson, I will use virtual instrument mobile apps to demonstrate timbres of particular instruments.

Researcher: Which software functions have you used for each type of software applied in teaching?

Joyce: For example, I will use music software to prepare a song which is related to a particular unit of the music lessons. Then I ask students to write lyrics for the music.

Researcher: What additional functions do you wish the software to have to meet your needs?

Joyce: I would like the music notation software to automatically generate *so fa* name (solfeggio) from the melody on the score by one simple click. This would be very helpful for students to learn singing.

Appendix P

Interview Transcript: Suki (pseudo name)

Researcher: Which particular brands of music software you have used?

Suki: Logic, Pro Tools, Cool Edit, GarageBand, Sibelius, and I don't like Finale.

Researcher: For each type of software, which one you think is better and why?

Suki: Sibelius is more user-friendly than Finale. Logic Pro is the best to me in audio editing, because I used to use MacBook and I don't have Pro Tools in my MacBook.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials? How is it being used?

Suki: Mostly to prepare class materials. For example, make some music clips to demonstrate particular music styles or instruments to students, or create some background music track so that students can add some creative things on the track such as rhythmic clapping. Notation software is used to prepare music exercise sheets. It's quite difficult for primary school students to handle notation software to create something on their own.

Researcher: Which software functions have you used for each type of software applied in teaching?

Suki: I use those primary functions that the software serves. For example, notation software is used to notate music scores, and GarageBand to make loop-based music. I know there are other functions, but those do not help what I want to do.

Researcher: What additional functions do you wish the software to have to meet your needs?

Suki: For educational purpose, I think music software should provide more templates. For example, templates to guide teachers to teach creativity – because teachers are busy, in those circumstances if software can help then it would be good. It would be good if the music notation software can accurately 'record' what I am playing on the MIDI keyboard.

Appendix Q

Interview Transcript: Galilee (pseudo name)

Researcher: Which particular brands of music software you have used?

Galilee: Adobe Audition.

Researcher: Have you tried other brands of audio editing software?

Galilee: Sibelius…maybe I just tell what software I have used. I use Sibelius for score typing, not actually during my teaching. Ladida, which is an iPhone apps. You sing something, and then it can tune the pitch for you and also make some background music for your singing to make it kind of stylistic.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Galilee: Most of the time it is for me to prepare class materials. I have also taught students to use Audition to remove the vocal part from songs.

Researcher: Which software functions have you used for each type of software applied in teaching?

Galilee: Not much other functions, I use the software because it serves a very particular function that I need.

Researcher: What additional functions do you wish the software to have to meet your needs?

Galilee: I want software that can notate the melody that I sing through the microphone. Sometimes students can sing the melody but have no idea how to notate it. Such a function could help.

Appendix R

Interview Transcript: Tony (pseudo name)

Researcher: Which particular brands of music software have you used?

Tony: Overture.

Researcher: Have you tried other brands such as Sibelius?

Tony: I have tried Overture and Finale for music notation software. Since the computers in my school do not come with Finale, I use Overture most of the time as it is free and I can install it everywhere. My tasks are simple, so Overture is sufficient for my needs.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Tony: I use it to make work sheets. Score typing.

Researcher: Which software functions have you used for each type of software applied in teaching?

Tony: I play the score notated, does it count? It is to make sure I am not typing any wrong notes.

Researcher: What additional functions do you wish the software to have to meet your needs?

Tony: Software that is interactive…for example, software that can assess students' composition. It can basically identify the wrong rhythms, or wrong notes from a given chord.

Appendix S

Interview Transcript: Phoebe (pseudo name)

Researcher: Which particular brands of music software have you used?

Phoebe: I use Finale and overture. Finale is better because the output is clear and tidy. It's functions are better.

Researcher: How is the software used? Do you use the music software during music lessons?

Phoebe: I use it to prepare class materials such as worksheets, and virtual piano apps for self-reference.

Researcher: Do you use functions other than score typing in Finale, such as playback?

Phoebe: No.

Researcher: What additional functions do you wish the software to have to meet your needs?

Phoebe: Hmm….not much.

Appendix T

Interview Transcript: Esther (pseudo name)

Researcher: Which particular brands of music you have used?

Esther: Finale.

Researcher: Have you tried other notation software?

Esther: Yes, I have used Sibelius also.

Researcher: Which one you think is better and why?

Esther: Finale, because I have it in my home, so I keep using it.

Researcher: Which mobile apps do you use?

Esther: Metronome, virtual piano. But it's for my private piano students.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Esther: To prepare teaching materials. But they know how to use because previous music teachers have taught them Overture.

Researcher: Do you use functions other than score typing?

Esther: I export the mp3 and use it for preparing class materials also.

Researcher: What additional functions do you wish the software to have to meet your needs?

Esther: I want to type percussion scores with it. Probably it comes with that function, but it is just because I don't know how to do this, right? I also don't know how to make music exercise sheet too.

Appendix U

Interview Transcript: Alice (pseudo name)

Researcher: Which particular brands of music you have used?

Alice: I use Sibelius to type music scores and Audacity to edit music files.

Researcher: Have you tried other brands of notation or audio editing software?

Alice: Yes, I tried Overture before. Actually it depends on what I can download at school computers. I used Logic Pro before, but it seems not so popular in school.

Researcher: Why do you prefer Sibelius but not Overture?

Alice: The score I typed looks tidier than that of Overture.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Alice: Mostly for preparing class material. I still cannot handle the software well to use it during the music lessons, and it also depends on whether the students have the same software or not. It's better if I can have some of the music lessons in computer room, but I am not good at computers. I use Audacity to mix different songs together so as to demonstrate the concept of tonality, or split part of the song for other demonstration purpose.

Researcher: What additional functions do you wish the software to have to meet your needs?

Alice: I don't know how to change the key during the notation process, but I think it's just my problem. The noise removal function of Audacity is not that good enough. I know there is some other software that can correct the tuning of a voice which is originally out of pitch, but Audacity does not have this function.

Appendix V

Interview Transcript: Mandy (pseudo name)

Researcher: Which particular brands of music software have you used?

Mandy: I used Noteflight, because it's free and do not need to install (Noteflight is a web-based notation software).

Researcher: Have you tried other brands of notation software?

Mandy: Yes, I also used Sibelius.

Researcher: Why do you prefer Noteflight rather than Overture?

Mandy: It's because Noteflight is free, and school cannot provide other software for me.

Researcher: How is the software used? Do you use the software during the lesson or just use it for preparing class materials?

Mandy: I use it both for preparing class materials and during the class. It's for me to prepare music worksheet and to teach students how to read scores when I teach them how to play the recorders.

Researcher: Which software functions have you used for each type of software applied in teaching?

Mandy: It's web-based software that I can only notate and print. That's all I need.

Researcher: What additional functions do you wish the software to have to meet your needs?

Mandy: Noteflight does not have a comprehensive sound library that many instruments are missing, and the sound quality is bad. It's bad enough that I dare not to generate audio sample from it.

Appendix W

Interview Transcript: Sharon (pseudo name)

Researcher: Which particular brands of software you have used?

Sharon: I use Sibelius and Finale. Mostly I use Finale, because the arrangement I have done can be used in SmartMusic.

Researcher: So Finale is linked to SmartMusic?

Sharon: Yes.

Researcher: How about audio editing software?

Sharon: Nothing right now actually, it's the past. I used Audacity the most.

Researcher: How about theory & aural training software?

Sharon: I am using SmartMusic currently.

Researcher: How is the software used?

Sharon: For students to practice at home. They can submit assignment through SmartMusic.

Researcher: Do you use other music software during the class?

Sharon: I use GarageBand and Logic mostly to record and make accompaniment.

Researcher: Which software functions have you used?

Sharon: For performance assessment. It would be good if music software could assess the students' performance. Have you used SmartMusic before? It will listen to a person's performance and then it will show whether you play the right note at the right time. But it doesn't assess how long your notes are. If you play something wrong, it will tell you. But it does not assess how long the notes you play are. It only assesses the first beat of any note on the score. So you could, for example, play a whole note, which is actually one beat, it will still show you are correct.

Researcher: What additional functions do you wish the software to have to meet your needs?

Sharon: It would be great if the software for performance assessment and evaluation can let the students know how to improve their skills instead of only detecting errors.

Appendix X

Interview Transcript: Dr Andrew King

Researcher: What is the core content of the programme BA(Hons) Creative Music Technology of the University of Hull?

King: Four main strains of that particular curriculum: performance, composition, musicology, and music technology. Technology is now better with the curriculum because it attempts to underpin the entire curriculum structure. We do a wide of things varying from live sound reinforcement to digital instrument design that looks at things that students design their own instruments using MAX/MSP, Objective C which brings both the digital world and the analog world. They also do studio production like surround sound recording, field recording. But we found important that they should have musicological context. So throughout the programme there are musicology courses that they study avant-garde music as well as other aspects such as song writing. There are lots of programmes in the UK across the 10 years which quite similar to this one. But they tended to fall into one of the category – music departments offering music technology or music & technology programme, or engineering departments focusing on scientific aspects such as digital signal processing.

Researcher: What hardware and software do the students use?

King: We were working on many different software platforms, and we recently narrowed it down to selected programmes that we use such as Pro Tools, Sibelius, Cubase, specialist programme such as MAX/MSP and Csound. We gradually replace Csound with MAX/MSP, which is an objective, graphic based version Csound.

What factors affect the change of music software that taught throughout the programme?

King: It is partially student-driven. Some of the software platforms such as Pro Tools are what the students are asking for. That reflects what's going on in the music industry. We try to teach the principles through the software, but also consider what our students want and are aware of as well. It is partially market-driven but what we try to give them is a broad range of software platforms so that they can understand the principles of the software and be adaptive to any studio environment, navigating around those systems. I did a study a couple of years ago that looked at the difference between analog and digital recording techniques. One of the thing we do in the first 12 weeks is we don't let them use the computer. They got to use all the analog equipments. So they learnt the principles of signal flow, signal route, signal monitoring without looking at the computer screen. I separated the students into two groups – one using analog mixing studio and the other using digital mixing studio. One thing that I found was the misunderstanding in the digital studio because it can be a lot more 'forgiving' in aspects like using reverbs in inserts, which are rarely done in

the analogue world. If you do it in the analogue mixing studio you can hear it is incorrect but the software will fix it for you in the digital mixing studio.
Same as Sibelius. It will highlight in red for you if you are writing out of the range. People now write complex clarinet parts that cross the (clarinet) bridge, which is unplayable to a clarinet player, but Sibelius could play it for you.

Researcher: What are the hardware and software inside the studio of the University of Hull?

King: Pro Tools, Sibelius, Analog and digital mixing and recording studio. Compressors, noise gates, pre-amps, a variety of microphones for different purposes. I can give you the full list later.

Researcher: What is the potential future of music software in teaching/ learning/ assessment?

King: I think the way it is going is towards the design. To me the real cutting-edge in terms of the software seems to be the musicians bringing their own ideas to design their own instruments – more user driven in a sense.

Researcher: Is it a problem that the music software developers do not consider what the musicians' needs? Or they just don't know what the musicians' needs are?

King: It is always better to listen to musicians. Software developers such as Sibelius are very good at attending music education conferences, talking to music teachers to see what they want. Cubase, on the other hand, also has plug-ins for teachers, but is aimed very much at pre-university level and doesn't always map onto what's going on in the national music curriculum. I think it is already better than it used to be but features such as notation I would never suggest students to use it in Cubase. Pro Tools also do MIDI these days but not very well in my opinion. Probably I would suggest Cubase for MIDI because it seems to be the standard, or Logic Pro. But I don't think any of those packages can touch at music publishing, that is, notation for publishable standard. I know Cubase is trying to do all three but fail to do the best of all. Music technology is a small industry in UK, the big budget for research and development you get are for more commercially viable programmes. One of the struggle in the procedural aspects of using the software is that students acquire the software from somewhere, i.e. downloading from somewhere, perhaps illegal, and play around with the new piece of software. But they are not experienced enough as a sound engineer. They do not have the perspective and get lost when using the software. Some of the software packages to me seem to be over-complicated. These programmes need to be simple. They got to be intuitive. There is a huge enthusiasm that the students use the software and technology to design their own music, but the teachers can't deliver it. My generation of teachers really tends to struggle about using any of the software. They know what they want to do but cannot get pass those actual procedures to achieve what they want. They can't acquire the support. If you talk to any of the studio shop that sell music systems and software to schools, they tend to find there may be a nice radio studio in the school with mixing desk, computers and other porting equipments, but no body knows how to use it. There may be one that knows how to use it sitting there 6 months but now gone. And the new music teacher comes in and again plays the flute and piano again. We almost seem to be at the stage of the moment where the pendulum swings back to the more traditional models of music

education. There was a huge enthusiasm for technology over the last 10 years, and now we are switching back to more traditional methods. We actually noticed that the type of programmes the students wanted to do, for example the music technology programmes that we run here. About 7 to 8 years ago we used to have about 500 applications for 50 places, whereas now we probably down to about half that number. I would say this is a trend, because the traditional BA (Hons) Music programme in the University of Hull 5 or 6 years ago it was really struggling to get students to study, but now their applications are up, more students apply for that programme again. So it seems to swing another way at this moment. May be this is affected by the economy. Students are paying 3000 pounds a year at the moment, but from September 2012, they have to pay 9000 pounds per year due to the change of policy. A lot of students going for music technology programmes thinking they are going to career path as sound engineering. We don't run a sound engineering programme per se, the industry of sound engineering is actually shrinking, getting smaller and smaller. So that's partially driven by economy.

Researcher: Is there any limitation of music software that is not favorable for classroom music education?

King: For example, Sibelius, is far easier from the music teachers' perspective, as they understand the music drama of the programme. From the students' perspective the main barrier is the lack of grounding in music theory. Not knowing about orchestration, not knowing about harmony. Perhaps too much emphasis is placed on the technological skills. Students are very creative, bring a lot of creativity to the programme. But with software like Sibelius, the dangerous thing is that if you do not have grounding of music theory, it is just like 'garbage in garbage out'. I think in terms of programmes like Cubase it's very difficult for the teachers to learn how to use these sorts of programmes, and again for the students, if you do not have the theoretical understanding of audio engineering, you can be get some results but not getting the best. I think we are moving towards a model that the technology just got to be the underpinning of the music curriculum. We don't necessarily have to have courses called music technology. I can see in next 10 to 15 years, technology will just become any strain of bachelor music, in the same way as composition and performance, and we don't have BA musicology, we don't have BA composition, we may have that in master-level, postgraduate may do MA in composition, or MA in performance, or MA in musicology, or MA in music analysis, something like that. I can see technology will go in to same way, I can see technology is just becoming a general underpinning to musicological courses, performance courses, compositions courses. Technology does exist, but it seems to emphasising more on the procedural elements. So we may find technology with an engineering department in the US, places like MIT. Perhaps not that much in the more mainstream music higher education.

Researcher: From the teachers' perspective, what can be done to facilitate the music software development for the sake of music education?

King: What ideally will happen is the community practice. There were pocket of good practices, but the local educational authorities don't seem to link well together. I have been involved in delivering some sessions where a local educational authority offers for a full day, I made sure that all the teachers attending knew how to use Cubase. Then they went back to the classroom, they forgot what they learnt. They were lack of

support. There should be more community practice to link different school together. The more you use the technology, the more you familiar with it. A couple generations of people are still working in music education, but they have not grown up with technology. There are great stuff doing outside the school, where the students use the software, designing their own instruments, things like that, but when it turns into the music classroom, the teacher doesn't know where to start. The enthusiasm has lost. So you got unhappy teachers because they can't deliver music technology. So they structure the music lessons in a more traditional way which produces unhappy students because what they actually want is how to use the technology to create music. There should also be a balance, making sure that the general music skills are expected to have in music lessons, but also technological awareness is also important as well. My experience of last 25 years is that the principles of sound recording have not changed. In terms of equipment that we use, you may have a software patch which replaces the physical rack, but the principle of signal processing is the same as 25 years ago. The only one thing that you can do in the computer that you cannot do in the analog studio is sample rate conversion. The technology nowadays is same as those 25 years ago, but just repacked. There are new things like MAX/MSP, that's new. Things like Cubase or Pro Tools are just speeding up the works we have been doing for 25 years, faster and more efficient.

Researcher: Is the music software for assessment purpose just a new trend or a sustainable future of music technology in education?

King: I think that would probably continue to grow. It almost sounds like an intelligent tuition system that is trying to give feedback to the performance. I think that kind of software is quite reliable actually giving feedback outside the actual music performance lessons. The music teacher can help the students to interpret their music performance. The difficulty with computers is giving meaningful feedback. We can always get data from the software, but how to interpret the data is another thing. This type of software probably will become valuable. I know a lot of academies still are not convinced that is anything better than the traditional way of teaching. But I think if those software are used appropriately, I can see how that could be a great benefit.

Appendix Y

Interview Transcript: Dr Evangelos Himonides

Researcher: How is the music programmes offered in the institute of Education?

Himonides: The IoE only offers master-level degrees. From the latest report, the MA (Music Education) is the only music education master degree in the UK.

Researcher: What is the module 'Music Technology in Education (distance learning)' about?

Himonides: The module is created about 6 years ago, when the need was identified to create something that focus on the educational perspectives of music, technology, education – The inter-relationship between music, technology and education per se. I identified 6 years ago there is no course in the world – because I have done a global market research – that was addressing educational perspectives and music and technology. Postgraduate level courses we are not talking about communicating or teaching people with particular heuristics of a particular issue, we are developing critical thinking. What we are trying to see in this module is the world of technology within music and education, becoming critical without practices and the employment of technology in educational setting, being able to evaluate the effectiveness or not of particular technology or tools within the classroom or educational context. And this course is available via distance learning as well. It is not compulsory.

Researcher: Is there any hardware or software used in this module?

Himonides: It is influenced by my ethics as well as practices. I combine music, technology and education. And because of my involvement in computer science, I have developed my own particular ethics and morals about the use of software and hardware. One of the things that I have been writing about is the need to separate heuristics and learning how to use particular tools or software, and what can be done with different kinds of tools and software. This particular course, I do not enforce the use and investigation of any particular platform. Everything we teach in this course has been designed so that it can be delivered using open source materials, trial-ware or shareware. There are plenty platforms that we might use and address, for example, we might use ripper, introducing people to sequencing. We might use digital analysis tools, real-time digital signal processing. Anything that can help people to visualise, or have different metaphors of sound visualisation. We also invite people to bring their own suggestion of software, and try to assess how effective or how more effectively this kind of software can be used in their own context.

Researcher: Is the music technology course more about contextualising or conceptualising?

Himonides: Exactly. It is more about being critical about particular things, not so much teaching people and providing empirical knowledge. This is mainly because of the character of a course under the umbrella of postgraduate level. Many postgraduate students face – particular the beginning of this course – they cannot make that manual-leaded, they wanted to have a much more dialectic experience which is not postgraduate level is about. Different realities exist in different educational context of different countries. And it's interesting to see that particular things are not the same in different countries – for example, the expectation of the tutor or professor being the sourer and the student being the appetent – something that you see in many countries. In the UK, there is a very clear shift from the appetent and sourer model. Especially in the postgraduate level, there is a non-dietetic component. There is a clear focus on the need for developing critical thinking. The content of the programme is not for you to learn new things. It's for you to develop critical thinking.

Researcher: Are there any special things that you can see from those students in the music technology course, such as their differences in expectation?

Himonides: People do often have issues about using a tool. Especially in the field of music education, there are many people that are very traditional, one might say non-technical. We are as much technological by default as we are musical by default. You will have many people that perceive themselves or having attitudes of digital immigrant. You will find many people focus on how to use particular software, treating it as a kind of blackbox. You can perform an analysis of those journals – IJME, BJME, MER – all the articles that having something to do with technology, they treat technology as if it is a piece of drug. The assumption is the important thing is to have the technology, to be the owner of the technology. And there is an assumption that every people are using the software in the same way. What is the need and what is not the need in using the software in such particular way are always absent. If you replace the term ICT or technology or software with any drug, you have yourself a valid paper. There are a lot of academies taking advances of that in order to promote their own interests. They do a lot of how-do lists and do a lot of blogging to the digital immigrants – 'You don't have to think about it, just to do this'. Many people are concerned about how to use a particular tool and they do not have an understanding about why they need to use that tool, or how things are going to change, what is the value-added, how are you going to make music education better, or facilitate better musicians. They focus on the actual tool. Many times you see that people don't want to think. They want ready-made, guaranteed results. I have never come across any creative writing programmes that teach you how to use Microsoft Word. Microsoft Word is just one year younger than MIDI. But we still treat MIDI as amazing tool that was brought into the world of music technology. Educators specialising in technology still have very unclear understanding of what MIDI is about. They have confusion between sound and MIDI. They are not in the position to answer simple things like why the MIDI file is so small for example, or why I play MIDI file in different computer it sounds differently. That's because of the concentration and number if digital immigrants and those digital natives have been taking advances of that, driving the place of music education to a place that I am not very happy with. Of course you need to understand how to use software, but just like the Chinese proverb says it is always better to show them how to fish than to catch a fish and give it to them. I want my students to learn that you can align particular

information in time, and what are the implications of doing something like that in the classroom.

Researcher: What is the potential future of music software in education?

Himonides: It is the focus on communication. This is a multi-dimensional issue. The discourse between educator, user, software designer, software developer and the music industry need to become much more open. At this model there is an unclear balance and unclear relationship between those. You cannot have a particular platform or software drive education, having music education needed to fit the particular platform or software. I see the future of software being completely open platform, completely extendable. You can use an XML base where any software can speak to any other software if they do not have to share deliverables, exchange meaningful information about what the product is all about. All the information needs be accessible to all the users in a meaningful way, communicated probably driven by the existing or future technology for social networking. At the moment the assessment is a huge issue because of the lack of communication between different strains, channels and maternities.
Any kind of collaboration would have problems, strengths, and allowance of creation in discourse, and may also facilitate something that can become very beautiful, meaningful, and important to the field. You will see that in most of the cases, issues and problems are created when people are narrow-minded. If you see a thing in a meta-prospective, music educators don't just want to teach music, they want to develop people that preassembly would be the audience of the software developers. The music industry would sell to the future listeners or the future makers.
Now every person can have their own computer, their own recording studio. The music software developers are no longer speaking to dedicated persons. They are speaking to the whole world.

Researcher: Is there any limitation of music software that is not favorable to music education?

Himonides: I don't believe any tool has inheriting problems. It is how we will use that tool. The software is there in order to solve a problem. Of course you would have an interface that is better than the other interface, or utilities that can do more than the others. I have an assumption that when someone launches software in a particular field, for example music technology, they want to provide something that is actually doing something. For me the important thing is how to build the house, but not [choosing] which hammer to use. Of course it will make my life easier if I use suitable software, it is an own fact. But for me the important thing is first to understand and second to communicate to my own students and other music educators what potential is, what can be done, what needs to be done, what could be done, what would be nice, if we have the position to do. If I am an educator I want to assess a composition, I don't want to just see the final score, if this is the case, why use Sibelius? If I want to develop my students, help them become better composers, I want to see the whole process of how they do it. It is difficult to share a global or universal database that can accept any kind of deliverables. I don't have the software or device to play what you have done, I should be able to look and rationalise and understand what you have done. There is no issue about copyright, software protection. It is all about communication, both for music educators and computer

scientists. The principle is we need to realise we are not focusing on a very few people that we call the leaders and everyone else are followers. Everyone is a leader and we want to develop everyone as a leader. The discussion cannot be lead by software developers because mostly the software developers don't have a clue about the real world. Software developers don't think like their targeted users. They will not make mistakes when using software. This is one of the problems in software development. Because the thing will become something that was perceived by a group of people that are completely outside the context that they want to target the software to. Sibelius used to do public consultations with educators. Sibelius has a very strong educational section. They discarded the moment they became part of AVID. They are going to pay for that. Because it will be reflected on the actual product they launch. That's why Groovy was so successful and actually so intuitive because they have spent hours and hours working with educators that knew what children these ages want to see. But it goes another way as well. I am not saying that educators are the voices need to be empowered. In many cases educators may be stuck in the technological world. That's why you would have educators nowadays in academia designing software for children that looks exactly like the textbook they had when they were 5 years old. Look completely monotonic, with interfaces that only they can understand where the 5-year-old children nowadays use interfaces are 10 times more complex than the whole control room of a starship enterprise. If you look at the modern game, then look at the software that have been developed for the same age groups by educators that do not have any understanding of design and modern human-computer interaction that is other way round. That's why I am saying that all of these groups have mediated by the business because business has to investigate into all these discussions. They need to have equal voices and participate in the discussion. Otherwise you would come up with either inappropriate or non-effective or non-meaningful designs. Things will happen, we still have music, and we still have music education. We just need to know what their role is, what their potential is, what the power is, and how wonderful it would be if that discourse begins at that kind of level. I don't need software that has tracks and channels. There are plenty of them. If I am not happy with them, I can sit down and write it. We need something that would enable people to become better in whatever they choose, either in music or education or in appreciating or in communicating music, or in editing, archiving, whatever.

Appendix Z

Interview Transcript: Dr Fred J. Rees

Researcher: What is being taught in the programmes MSc in Music Technology and BSc in Music Technology at the Indiana University – Purdue University Indianapolis?

We have a two year musicianship sequence, in the traditional American undergraduate programme, you have two years of music theory, ear training, piano class, and aural training. Usually taught by different people. There is very little connection between what they do, just to satisfy individual course requirements. We have taken those separate courses, make them into one big course. So a team taught by three faculty members who have strength in different aspects. So they work with students 90 minutes a day, 5 days a week for two years to develop their musicianship. They build it in the workstation, so we use music technology to help them to build their musicianship at the first two years. Then we have three courses, Music Technology I, II & III, which teach them how to use the technology, for recording, for musical production, for performance, adapting certain software, and they finish at Music Technology III with to do a project. They take other courses as well, such as music lessons, playing in the ensemble, and university courses, language, history, maths, something like that. But then they finish with what we call the Capstone, that is a big project that synthesises all their experience in their music technology courses. We offer elective courses and right now we are doing one for those who are interested in studio recording. So they go to the professional recording studio and work with the environment.

Researcher: What software is used?

Sonar, Logic, Pro Tools, Reason, Finale, Ableton Live.

Researcher: What factors affect your choice of music software being use?

We used to use Sibelius. But then our broad of advisors we have the head of MakeMusic. And so we switch over to Finale. We have been using Sonar because of the instructor, he prefer Sonar so that's what he used. But we have changed that now because Logic and Pro Tools are the industry standards. And we want students to be able to familiar with them before they go out and work. The decision is generally made by what's going on in the industry. There are some differences between Sonar, Pro Tools and Logic, but not a lot. But the reality was that the students only used Sonar for 4 years, and then they went out and figured out that they had to use Pro Tools, and they hadn't seen it before. The transfer of knowledge would be a problem to them.

Researcher: Do you agree that music software development has not yet considered the pedagogical needs?

I have they all have their limitations. If you go back 20 years ago, we have drill- and practice-based programmes, for everything from ear training, sight reading, to music theory. You take the test, it will say 'yes' or 'no', 'correct' or 'incorrect', and then it gives you a score. But that's all it did. They were cheap. And then we have more sophisticated software, things like Auralia. I agree with you and I should tell you my fellow professor Dr. David Peter is actually studying software for the iPad that has music educational value. Because he owns the company that used to produce those practice-based programmes for a decade.

Researcher: What can the iPad to do for us?

It's cheaper than a computer, and because it's icon driven I think it is more convenient. If you are looking at an orchestral score, that's a problem. But if you can connect the iPad to a bigger screen then you can look at the full score. But that's the limit right now. There is something similar right now, the MusicPad, which can show the full score or part score. When you click the foot pedal, it will turn the page. But it's expensive, and single-functioned. There is some distinctive limits to the iPad, not able to connect to a computer, not able to store information. But it's convenient, light, graphic is good, good sound quality.

Researcher: There are many music educational software nowadays, but most are drill-based or wiki-based. What do you see the future of software – what can they do for music education?

Researcher: I would like to see good ear training and sight reading exercises. The students can do it away from the classroom. To me, we can use iPad in the class, and can also use it to practice, to read, to learn, to compose away from the class. It is a classroom itself. If we create software that instructs the students, give them samples, students come to the classroom with some questions, you use the time at the class to solve the problem. We use the time at the class to give feedback, and to perform based on whatever the tasks are. Students can learn from these kinds of mobile technologies if they are put together in the right way. We only process some of the information that we can read, like in a book. Interactivity is the thing that makes it work. Everybody is interacting through facebook, texting, why don't we take advantages of this interactivity? So when you talk about new music software for education, they have to have high level of interactivity. This is what we called the adaptive learning, which has to do with computer being able to process your answers and pick up patterns. Let's say it is a listening lesson, or a singing lesson, you sing to the iPad, and it shows you the melody line, and you don't seem to sing correctly, it can pick up the errors, but not just saying you didn't do it well and ask you to do again. It will create tasks that point to your weakness and let you practice to improve your weakness. The programme itself is teaching the student, by giving feedback. The other thing that it needs to do is not just saying you got it wrong. But why did you get it wrong. One of the things that we really need which might make the development of music application better is a master programme that identify all those attributes - listening to the right notes, singing the right notes, doing something in the right meter – and give them values. So that you can attach applications to it. Just like the idea of the MIDI protocol. You create a piece of hardware or software that can do certain things – detecting the pitches, rhythm, timbre – and it will just do that regardless of what programme you are working with. What we call a universal protocol.

Researcher: But what actually is the protocol is? What is the real product?

It depends. For example, if I am teaching piano class. I want to teach scale, or chord progression. The protocol is already there, and all I need to do is to attach the musical content to it, so that it can be seen and heard. Just like the machine language. You have to have the core, and then you have the programming languages, and then you have the applications, on top of that.

Researcher: What is the potential future of music software in education?

There are several reasons why teachers don't use the technology. For people in my age, they have been teaching in the traditional way for so long and they could not change. They don't need the technology for their situation. But I think the younger people, as long as they think the technology is simple, they just use it. You don't have to worry about the music. So I think as the computer become more and more powerful, it can embed more software that is easy to use, and the kids will just use it. I think there are bigger challenges, for example, some people who do not have traditional music education, and they use GarageBand to make loops and hip hop and so on, and that become the future of music. 50 years ago, if you want to make music, you have to learn the instrument. It takes you several years to really play it well. Now with synthesisers, you can make music by only clicking buttons, you don't even need to learn how to play keyboard. And today, with the looping programmes, such as GarageBand, you can create songs without any music theory or reading music. More and more people can engage in making music, but the learning of music, technique or content, that still requires training. The biggest challenge I can see at least in my culture is they look for the easy way out. If they don't have to learn, they won't. And then I also see the downside of technology, and because there are so many distraction and directions to go, they feel they don't have to profession in any area. And I see this is a trap. Well hopefully that will be the teachers' job; we don't want students to write 100 songs which all sound the same. When we get to the point that students think that 'this is not good enough', and 'I want to know more', then that's the chance we can teach them. Unfortunately we created a lazy world, people nowadays just ask 'what's the answer?', well you have to work through the problem to get to the answer. The same problem happens in music. Our technology gives us the answer that we don't have to work for it.

Appendix AA

Interview Transcript: Mr Matti Ruippo

Researcher: What are the music programmes in the Tampere University of Applied Sciences?

Ruippo: You can read the curriculum from the university's website, the curriculum on the website is still valid. There is an option in the School of Music in my university for music technology pedagogue.

Researcher: So there are a number of music programmes in your school?

Ruippo: Yes, seven altogether.

Researcher: So what's so special about the music technology pedagogue?

Ruippo: The programme has acquired validity for training teachers – not for elementary schools or secondary schools but music school in the area of extracurricular music studies for youngsters, for vocational schooling systems.

Researcher: Is it one kind of music teacher training programme?

Ruippo: Yes. Please remind me to send you the English link about the curriculum.

Researcher: What kind of software do you use or teach in the programme?

Ruippo: We do studio recording, in which we mainly use Pro Tools. We also use Logic, Cubase – but this is not the main programme we are studying. There are Reason, Ableton Live. For notation, Sibelius. We have Linux based software too. Then quite a lot of software for pedagogical purposes because students they are going to teach in schools. Also some online resources. Then GarageBand, and iMovie.

Researcher: What factors affecting the choice of music software?

Ruippo: For example, Sibelius, it is the leading software being used. Pro Tools is being used because it is the industry standard. We are certified training center for Apple Logic, I am a certified Logic trainer, so Logic is being used.

Researcher: There are many music educational software, most of them are drill-based; some is wiki-based. What do you see about the future of music educational software?

Ruippo: Interesting thing that has happened already is on your mobile there are several hundred small applications. It's quite difficult for me to see that there will be some kind of learn-all-music in one programme. For instance, the majority of the music educational software, they are in English. This is a little bit difficult for people

in Finland to use who are speaking Finnish. Another thing from the perspective of music teacher, the problem is that: I have some kinds of visions about music education, I am teaching that way. Music software programme developers, they are thinking differently. So for me as a music teacher, it's easier to take some parts of the programme and use of them, but I have not seen any learn-all-music programme. I can only say, hey, this exercise is good, do that. There are many parts that I would use. I don't need them. So, as a matter of fact, I think those small one-task programmes are more beneficial in that meaning. They do one thing, they do it well, that's enough for me. If computer is so good that it can replace the teacher, then he or she shouldn't teach anything.

Researcher: Is there any limitation of music software that is no favorable for music education?

Ruippo: It's again the language problem, and also music cultural things like Chinese music which is totally different from the Western. I think that mainly it's the problem of how you use the programme. Teachers always tell their students, you press that button, that will happen, you press another, something will happen. But you don't tell the students why it is important. What is the meaning for the music? Teachers should be teaching music when they are teaching music software. If I take Sibelius as an example, there are a lot of notation themes. As a teacher I have to explain why it is important. What's the meaning for the music? Same thing when you have to use a mixer. What is this button for? For me, software is one part of my teaching, not the whole teaching. So the software is a tool, a mean, not an end.

Researcher: What is the future of music software in music education?

Ruippo: When the signal analysing system gets better, because right now if you have an audio, you put it in some music analysing programme, it can't analyse it very well. It only analyse the frequency, dynamics, and that's it. It doesn't know whether that singer is an oboe or a clarinet. It can't divide these kinds of things. During the years it will be some 'sunny' day, it's possible that computer can analyse music better. And that's a big leap for music education too. It's very difficult, but it is not impossible. We have been telling music teachers for a long time that please use technology in music lessons. They have to see good practices before they are convinced to use it. First, teachers have to know how to use music software in teaching, and then when they have this motivation, then you can also show them how the software works. Meaning first, then how-to.