Developing and Delivering a Software Internationalisation Subject

Tony Sahama, Chris Ho-Stuart and James M. Hogan

Faculty of Information Technology
Queensland University of Technology
GPO Box 2434, Brisbane 4001, Queensland

{t.sahama,c.hostuart,j.hogan}@qut.edu.au

Abstract

This paper describes the content and delivery of a software internationalisation subject (ITN677) that was developed for Master of Information Technology (MIT) students in the Faculty of Information Technology at Queensland University of Technology. This elective subject introduces students to the strategies, technologies, techniques and current development associated with this growing 'software development for the world' specialty area. Students learn what is involved in planning and managing a software internationalisation project as well as designing, building and using a software internationalisation application. Students also learn about how a software internationalisation project must fit into an over-all product localisation and globalisation that may include culturalisation, tailored system architectures, and reliance upon industry standards. In addition, students are exposed to the different software development techniques used by organizations in this arena and the perils and pitfalls of managing software internationalisation projects.

Keywords: MIT curriculum, Software internationalisation, localisation, globalisation.

1 Introduction

Over the last several years, interest in software internationalisation awareness and techniques has increased markedly. The spread of globalisation is analogous to, and indeed a consequence of, that of the internet. Thus, it cannot be safely ignored by developers. Indeed, the imperative is becoming stronger, with internet usage expected to increase significantly in the coming years in non-traditional markets such as the Asia-Pacific regions. The key to conducting business effectively in this environment lies in successful and cost-effective localisation – the systematic approach here termed internationalisation. Understanding internationalisation and localisation issues is essential for companies seeking to go global or who are already global, and thus for the graduates the employ. Our subject, Software Internationalisation (ITN677), introduces students to software internationalisation, localisation and Culturalisation paradigms. An appreciation of these issues is developed within an applied, technical context, with a strong emphasis on the practicalities of development. Students will also be able to consider how dissimilarities between countries in terms of their cultural, linguistic, legal and financial sectors affect the deployment of software onto international markets.

Many vendors have addressed, or partially addressed, these issues in the course of exporting software – catering for locale and more subtle cultural issues, customizing marketing efforts, and sometimes by developing new products and services more specific to the target environment. Successful software internationalisation, however, requires at least three ingredients to be in place:

(1) An adequate development environment – including programming language support;
(2) Maintenance of internationalisation resources and adequate tools;
(3) Access to quality expertise or skilled graduates.

The facilities required in points 1 and 2 have to some extent been delivered by the industry. The introduction and improvement of software internationalisation subjects satisfies the third criterion. A quick survey of our own showed few academic institutions around Australia have conducted software internationalisation subjects over the last two years, although tracking down schools and universities that offer this subject is difficult. Some institutions and universities may, in fact, be offering a software internationalisation subject (course), but this information may be hidden behind a special topic code or within seminars and workshops.

1.1 Subject History at QUT

Our decision to offer courses in software internationalisation - or 'software development for the world' - was motivated in part by the significance of the area for the industry, and by the tremendous range of languages and cultures represented within our student body. The subject was first taught as an elective in 2003 semester 1 within the Faculty of Information Technology, QUT. It will be offered as an elective subject at least once a year in the future. As an elective the subject is open for both masters students of the Faculty (MIT) and other streams with an interest in information technology across the university. The development of the elective was part of a Faculty strategic initiative in this area, with a particular intention to equip new graduates with these skills.

Two master's programmes are offered within the Faculty, catering to students with differing skill levels. One is a 'conversion' course for graduates without previous IT experience and the other is for those with a Bachelor's degree in IT. The revised master's courses - about to commence in 2004 - provide the flexibility of a wide range of electives for students who want to specialise in particular areas of interest. The MIT curriculum consists of programming languages such as Visual Basic (VB.NET), Java, C, C++ and C#, computer hardware, Operating Systems (Unix, Linux and Windows), databases (design and applications), system analysis and design, and networking essentials. Students are also required to take at least 4 electives related to their major area. At the undergraduate level, Bachelor of Information Technology (BIT) students and honours students, in addition to exposure to software development through undergraduate BIT courses and project management experiences, take undergraduate level courses in Software Engineering, Data Communications and Information Systems which include instruction in advanced software development theory and applications.

While students at both the undergraduate and graduate levels learn about the software development life cycle and application development as part of their curriculum requirements, we believe that a software internationalisation project involves a significant amount of new theory and novel practice, and thus warrants a course in its own right. Whereas traditional software engineering subjects are designed to give students an understanding of software development sufficient to support single locale systems, our specific application requirements focus on software internationalisation. In the software internationalisation subject, the idea is to create analytical, innovative and timely approaches to support the strategic application requirements of global deployment. In addition, the software internationalisation subject incorporates the idea that both operational and analytical applications should be an integral part of corporate data architectures and application development. This elective subject is intended for students interested in pursuing further applications in local and global software development, as well as for those students interested in understanding how to set up an over-all data and application strategy for an organisation in their own natural language.

1.2 Subject Objectives

The objectives of the software internationalisation subject offered at QUT are described as follows in the subject outline:

By the end of the course, students should be able to

1. define Software Internationalisation and the key elements involved in internationalisation
2. define Globalisation, Localisation and Culturalisation and explain the differences between these issues
3. assess the differences involved in internationalising, localising software versus online-help versus documentation and explain how they interrelate
4. critically compare and assess translation technologies
5. gain practice in the development of internationalised and localised software using the facilities available in the .NET environment
6. describe the implication of international contexts on cultural concerns, Graphical User Interfaces and the differences in the legal and financial sectors. Evaluate how differences in the international context, such as culture, language, law and finance, affect the design of Graphical User Interfaces and software in general
7. describe Internationalisation and Localisation project management and best practices
8. demonstrate awareness of social responsibilities by being able to assess and remedy the impact of poorly or well designed software on different groups of people

[together with a number of standard requirements pertaining to communication skills and team work.]

1.3 Subject Pre-requisites

As noted earlier, the internationalisation subject builds upon earlier experiences in software engineering. The stated prerequisites for this subject are:

- understanding of an object oriented programming language (basic programming paradigm)
- a good understanding of business needs and practices
- a background in some form of software development and management

Thus, students should have some significant software development experience, and a good understanding of software engineering and the software development life cycle prior to commencing this course. We have not at this point attempted wholesale integration of internationalisation content within our more introductory subjects.

1.4 Subject Content

We designed the content of the subject to be delivered over 13 weeks - assuming a 1 hour lecture and 2 hours of laboratory practicals per week. This subject extends student skills in software development, structured program design and implementation through a widely used commercially oriented programming language and development environment. Programming examples were drawn from typical industry applications. The subject begins with the evolution of software internationalisation and ends with an application-driven laboratory based examination. The exercises are designed to reinforce key concepts and to assist in the completion of the 2 major assignments and final examination. The topics listed below were delivered over 13 weeks with a number of guest lectures from the industry. To complement the lecture topics, students were assigned weekly practical activities in the lab as well as assignment related group

work that included reading, researching, discussion questions, problems and a short presentation. These activities are summarised in Tables 1 and 2 below.

Week	Lecture/Discussion Topics
1	Introduction to Internationalisation and Localisation: Definition of terms, Overview of the industry and its evolution costs, Purpose, and key results
2	Internationalisation and Localisation of Software: Translation, Locales and various character sets, Modularisation
3~4	Software Architecture and Internationalisation: Resource bundles, String management and context, Translation and its implications
5	Software Engineering and Internationalisation: Separation of concerns, Engineering workflows, project management, Testing and Quality Assurance
6~7	Documentation and Online Help: A Translation, Tools: benefits and limitations, Testing
8	Web and Web services: HTML Ver 4.0, XML as enabling technology, Testing
9	Guest Lecture
10	Standards and Tool Support: Why standards, ISO standards; W3C, LISA (Localization Industry Standards Association)
11	Further implication of international contexts: Cultural concerns, Graphics and user interfaces, Differences in law, finance, etc.
12	Guest Lecture
13	Review

Table 1: Summary of Subject content

Assessment Item	Description
1: (Assignment): 15% weighting Objectives items 3~10 Due during week-6	Documentation, Standards and Tools. Groups of two or three students must make an independent investigation of one of these aspects of internationalisation, and present their results and conclusions in a short class presentation.
2: (Group Project) 20% weighting Objectives items 1~3 Due during week-11	Software Engineering and Internationalisation. Groups of two or three students must develop a small application in the .NET framework demonstrating the aspects of Internationalisation taught in practical classes.
3: (Final Examination) 65% weighting Objectives items 1~5 and 6~10 During the Final Examination period	Final Examination (Laboratory based, application driven)

Table 2: Summary of Assessment Items

2 Practical Work

Given our stated focus on technical issues, practical work formed an essential component of the subject. Indeed, the final exam was conducted on-line in a laboratory session, and required students to use some of the tools and techniques that had been discussed.

Theoretical concepts are, of course, also of great importance, and so the exam did include a few short answer questions intended to measure how well students had assimilated some of the foundational concepts. Ultimately, however, this is a very practical and technical area, and the role of theory is to support effective practice.

Effective practical work in this area is not without its dilemmas, however. An application which is written to be independent of its locale works by accessing data describing local conventions in a standard form. This includes general conventions shared by all applications, such as number and date formats, and also application specification information such as translated strings, which must be generated for each new locale. The executable application, however, is independent of the particular data, and should run equally well in many different locales.

The most cost effective way to provide this style of independence is to rely on information which is provided in standard forms; and this means that applications are dependent on the platform. Internationalised applications can move smoothly from Japan to France; but they will not move so easily from Microsoft Windows to Unix, as the supporting API is different.

For practical work at a small scale suitable for a one semester subject, a platform had to be chosen, and the platform chosen was Microsoft .NET. The .NET platform provides an excellent globalisation interface, and it is itself an internationalised application. Similar levels of internationalisation support are of course available within java – and indeed were available earlier within java – but the decision in this case was influenced strongly by the availability of a standard IDE through Visual Studio and by recent advances in the Visual Basic specification which have greatly improved its viability as a teaching language. Visual Basic 7, which runs on the .NET platform, now provides standard object oriented features with a greatly improved match to good objected oriented theory. Moreover, it does so while retaining a simple and intuitive development interface, allowing rapid

prototyping of a graphical user interface with a highly professional appearance.

In the context of a software internationalisation subject, there was one unexpected advantage with this platform. A Visual Basic application is generally produced by alternating between a *design view*, where one arranges objects (controls) on a form; and a *code view*, where one writes the program code which handles the events and behaviour of the application.

Of course, laying out buttons and labels on a form is just an easy way to write the initialisation code that creates the various objects and gives them appropriate attributes. In Visual Basic 7 (VB.NET), this designer-generated code is no longer hidden from the user, but can be seen and even edited (if you dare). There are simple conventions for expanding and hiding the generated code.

It is possible to set up forms and attributes in such a way that the generated code already illustrates important aspects of internationalisation. That is, students are able to develop a very simple program, all without actually writing any Visual Basic code at all, and to see the effects of internationalisation applied to their design.

Of course, students went through a series of progressively more sophisticated examples; but the benefits of a gentle introduction with little coding required and attractive results was a great help for engaging their interest, and focusing their attention upon the internationalisation issues rather than wrestling with all the extraneous programming issues which are still a challenge to weaker students with more limited experience of coding.

2.1 Progressive Practical Work

Practical work in the subject centred on a simple application called the *Birthday Book*. We had been inspired by Nick Symmonds' book (Table 3, Textbooks Item 4), which illustrates many of the aspects of internationalisation by working through a single large example of a resource file editor. The *Birthday Book* application had some similarities with the editor, as both applications involved entering information though a *DataGrid* control. The *BirthdayBook* was a much simpler example. Students were also provided with an implementation of the Symmonds editor, which was very helpful as a model example of Visual Basic code.

The Birthday Book allows a user to enter and maintain information about the birthdays of their friends. Each week, students would make some new version or extension to the application, which illustrated a new concept in internationalisation or localisation. These concepts were also explained at a more theoretical level in the lectures.

One of the first exercises involved *localisation* of the application to fit a new culture. This was relatively straightforward, but time consuming. A subsequent exercise was to *internationalise* the application, so that it no longer had locale dependent features encoded within it. After this, a second *localisation* was performed, and students were able to appreciate how much faster and more efficient it is to move to a new locale when an application has been designed to support that process.

Subsequent weeks introduced many aspects of internationalisation, by adding new features to the application, each of which required special handling. One week involved saving and loading birthday data into a text file. At another time, students were required to write information in a form readable in the current locale. The following week, students had a good example of cultural dependence, when a file saved for one locale was loaded in a different locale, and consequently misinterpreted. This gave an excellent introduction to XML as a data storage method which is independent of locale; and was easy to perform since XML is the underlying data transfer method for .NET as well.

One difficulty with the progressive practical work is that a student could easily get the subconscious idea that .NET is the way to write internationalised software. The first assignment went a long way to helping students appreciate that internationalisation can (and must) be done when writing on many different platforms. The second assignment was a research project, in which students investigated literature concerning one aspect of internationalisation, and then reported back to the whole class in a lecture session set aside for that purpose. Many students investigated comparisons of Java and .NET techniques for internationalisation, and this helped students to appreciate that internationalisation is relevant for development on any platform.

2.2 Textbook Support

One of the challenges in teaching a software internationalisation course at the present time is that while many books are now available on the topic, the vast majority are trade books written for the specialist professional rather than the student. This means that the books often assume a familiarity with corporate business practices and some form of assumed knowledge of software internationalisation that many university students lack. These books also lack the exercises, discussion questions, cases, and other amenities usually found in academic textbooks to help guide the student (and the teacher) through the material. Finally, while many trade books provide an excellent conceptual picture of software internationalisation, many lack the specifics that students are hoping to acquire as to how one actually sets up a software internationalisation application development for the given region with appropriate and available tools and techniques. In practice, most trade books either deal with one issue or the other, creating a requirement for a number of texts for the subject. This was not economically feasible for the majority of students undertaking the subject and we prepared supplementary notes for respective lectures each week.

For the subject, we chose a few text books listed in the Table 3, that we felt came closest to the ideal textbook. However, there were no prescribed textbooks for this subject. Students were provided with relevant articles and notes on the topics covered. It is not our intention to review within this paper the range of text books listed.

Nevertheless, while no single text proved entirely appropriate, each of those mentioned covered some aspects of the material well and were a useful resource for the students.

Textbook Items	References
1	Luong, T. V, et al (1995). Internationalization: Developing Software for Global Markets, John Wiley. ISBN: 0-471-07661-9
2	Lustig, M. W. and Koester, J. (1993). Intercultural Competence – Interpersonal Communication Across Cultures, Harper Collins College. ISBN: 0-06-044129-1
3	O'Donnell, S. M. and Martin, S. (1994). Programming for the World: A Guide to Internationalization, Prentice Hall. ISBN: 0-13-722190-8
4	Symmonds, N. (2002). Internationalization and Localization Using Microsoft.NET, Apress. ISBN: 1-59059-002-3
5	Uren, E. et al (1993). Software Internationalization and Localization: An Introduction, Van Nostrand Reinhold. ISBN: 0-442-01498-8
6	Dr. International. (2002). Developing International Software. Microsoft Press. ISBN 0-735-61583-7

Table 3: Textbooks

Text book sources were supplemented in many cases by on-line materials and information, which students accessed more independently. Yet while there are a large number of internalisation web sites available, the vast majority are commercial in nature with information directed toward the buying public, with no release of technical details. In addition, many of the internet sites visited were poorly organised. To assist other course developers, we have used the references section to list the URLs of those sites which proved especially useful for accessing relevant information.

3 Approaches to Teaching and Learning

We managed to assemble a diverse collection of examples, exercises and projects drawn from many fields and designed to provide students with a chance to solve interesting, real-world problems. The laboratory exercises taught leading edge topics in the industry environment. The lab set-ups avoided arcane terminology and syntax specifications in favour of teaching by example. The programming code examples and the text emphasized good pedagogy. The emphasis placed on practical work was seen through the laboratory exercises, homework exercises and assignments. These teaching and learning tools are aimed at assisting students to complete major software internationalisation based assignments.

3.1 Feedback Mechanisms

The learning process was focused on real-world scenarios in industry, organised for presentation in a classroom environment. In order to meet this requirement, students were given complete working programs to demonstrate important concepts, and required to develop smaller applications using the same techniques. Assignments were structured according to industry standards and real-world application requirements. Marking focused on the depth of knowledge acquired, timely development and deployment of programming applications by using laboratory practices, theoretical knowledge acquired during lectures and additional readings. Upon completion of assignments, face-to-face, step-by-step constructive feedback on each assignment was provided. The final examination was assessed through the theoretical concepts covering the course and how those concepts were applied in practice.

3.2 Challenges in Design and Delivery

We faced some additional challenges besides the lack of software internationalisation text books and supplemental course materials when trying to teach this subject. One issue involves the changing subject matter. Because this area is still relatively new, the concepts continue to evolve. For example, there are debates on:

- XML standards and localisation
- Standards on locales
- Standards on on-line documentation and format.

Continuing advancement in software internationalisation standards, quality, portability and affordability are another source of change. Because of the evolving nature of the subject, an instructor should expect to revise at least some of the definitions and concepts taught from one semester to the next.

Another challenge is to incorporate some hands-on training so students obtain practical experience as well as a solid conceptual foundation. For realistic hands-on experience with software internationalisation paradigms, students need access to the appropriate software and real world problems. One of the most important pieces of software in software internationalisation is the .NET framework and VS.NET environment. We were successful in implementing hands-on practicals using VB.NET in the Faculty laboratory environment. The majority of the students who enrol in the subject have previously been exposed to Microsoft's .NET environment at some point in their studies, but this approach is being supplemented by the use of Linux and Oracle based software in upcoming revisions to the curriculum. Students may additionally wish to explore other software products that assist in application development, creation of locales and creating resources files as a part of the subject.

In addition we have been fortunate to have received support from two industry partners, the Brisbane offices

of Red Hat and Oracle. Guest speakers from each of these firms have interested students with real world problems and their hands-on experiences and both companies have been strong advocates for ongoing research and curriculum development in the area.

4 Student Reaction to Subject

At the end of semester 1, 2003, students filled out a voluntary questionnaire to provide feedback about the course. Because the subject (ITN677) was introduced for the first time in that semester, we expect the curriculum to undergo a number of development iterations before it reaches stability. As the subject was offered to MIT courses and with a mixture of student enrolments (Post Graduate Diploma and Honours) the collected responses were pooled. The response rate for the total cohort was 73% (n=29).

In reviewing the results, a high percentage of the master's level students (82%) found that the textbooks were not helpful. Most students (76%) felt that the subject was very successful and timely, but suggested that it needed more input from real world problems. Roughly half of the total cohort felt the pace of the subject was about right, although it was not surprising that some students (8%) found the pace to be slightly fast compared to other subjects in the Faculty. The majority (86%) said that they had learned something that they considered valuable and around 72% of students said they would recommend the course to a friend. These results suggest that the subject may also prove viable if offered in future semesters as an undergraduate elective subject.

More generally, students said they liked the assignments and group presentations that were hands-on, practical in nature, and gave them a realistic feel for how they would construct a software internationalisation project in the corporate environment. They also enjoyed the involvement of industry speakers and the real-life examples used during the lecture to illustrate the concepts being taught. Some students suggested that they would like to develop their skills further through a larger project.

5 Summary

In summary, the software internationalisation subject introduced this year represents new opportunities and avenues for the students within the Faculty's MIT programme. Our approach to software internationalisation gives students the opportunity to learn about architecture and application issues on a corporate scale, to practice the steps involved in the software internationalisation process, and to gain a greater understanding of application development, standards, and quality assurance as they apply to a software internationalisation project. Moreover, the approach has proven successful with students, and may in future be offered to advanced undergraduates as well. Incorporating the material within introductory subjects or repackaging the present offering for such an audience may prove more difficult, however. Strong familiarity with the software engineering lifecycle and some degree of maturity as a developer would seem to be essential prerequisites for such technically demanding content.

This paper attempted to describe both our experiences and our perceptions of the type of content and activities that should be covered in the subject. Some of the current challenges in offering the subject include some deficiencies in text book support, the changing nature of the topic and a lack of small scale real world problems suitable for use as hands-on exercises in the class room environment. Despite these problems, this subject offers students an important opportunity to explore further application development in the software internationalisation arena and to appreciate the value of a cohesive software development strategy for an organization.

6 Acknowledgments

The authors wish to thank the Faculty of Information Technology and especially the Dean, Prof. John Gough, and the Director of Research, Prof. Binh Pham, for their support and funding of this initiative. Our work has benefited considerably from the involvement of representatives from Oracle (Brett Hooker and John Richardson) and Red Hat (Paul Gampe). The authors wish to thank John Hynd for his help in preparing this paper, and to acknowledge the research assistance of Cindy Tong and Beth Hergenhan.

7 References

Useful Web Sites

1. http://www.i18ngurus.com
2. http://www.lisa.org
3. http://www.w3.org/International/
4. http://babelfish.altavista.com/