The Design and Implementation of an Enculturated Web-Based Intelligent Tutoring System for Computer Science Education

Phaedra Mohammed
Dept. Computer Science & Information Technology
The University of the West Indies
St. Augustine, Trinidad and Tobago
phaedra.mohammed@gmail.com

Permanand Mohan
Dept. Computer Science & Information Technology
The University of the West Indies
St. Augustine, Trinidad and Tobago
Permanand.Mohan@sta.uwi.edu

***Abstract*—Accommodating for learner diversity based on cultural backgrounds has not yet been a major personalisation focus until recently. Increasing numbers of Internet-ready devices have propelled e-Learning forward such that these deficiencies in cultural-awareness can no longer remain unattended. Despite being investigated from an instructional design standpoint, the enculturation of digital learning environments has largely been theorized, necessitating manual enculturation by experts, instructors and even students. Consequently, enculturated learning environments are limited in practice. In this paper, a preliminary design for building a web-based Intelligent Tutoring System (ITS) is described together with the features and intended functionality of the various components. This work contributes a practical approach that was implemented and evaluated using two concrete systems within the domain of Computer Science education. An analysis of the findings and empirical evidence reported in the study supports the viability of the approach taken and reveals that intelligent tutoring systems benefit from enculturation.**

Keywords-Intelligent Tutoring Systems; Culture; Computer Science Education; Software Design; Experimentation

I. INTRODUCTION

Interactive educational systems deliver instructional content to learners with the intention of providing a customised learning experience. This is achieved through personalization based on a variety of dimensions: learning styles, instructional objectives, devices used for content delivery and so on. Accommodating for learner diversity based on cultural backgrounds however has not yet been a major personalisation focus until recently. Increasing numbers of devices capable of accessing the Internet have propelled e-Learning forward such that deficiencies in cultural-awareness can no longer remain unattended. Educational content and online tools were originally considered to be more usable if they were designed without any culture-specific features. However, the development of culturally neutral content and tools is virtually impossible since cultural partiality pervades every design choice.

The design of user interfaces, the selection of teaching strategies, the format and content of the educational material all vary depending on the cultural background of the developers [7]. Subtle cultural influences seep into the final product and this can be counter-productive to learning when stereotypes and personal interpretations clash with the practices and beliefs of the students. So, by internationalizing or localizing these systems [12], certain users may be included and others left out thereby working against the goal of providing individualized instruction to any learner at any time. This happens largely because the cultural background of a learner plays a significant role in shaping his/her learning habits, and cultural appropriateness can no longer be treated as an optional personalisation dimension.

Despite being investigated from an instructional design standpoint, the enculturation of digital learning environments has largely been theorized, necessitating manual enculturation by experts, instructors and even students. Consequently, enculturated learning environments are limited in practice because many developers have shied away due to the complexity in reliably representing aspects of a particular culture [11, 2]. Young [12] points out that the dearth of culturally-aware ICT systems can also be attributed to the lack of guidance regarding the integration of culture-specific elements into present-day instructional design. Above all, the knowledge and processes for incorporating culture have not been clearly defined with automation in mind [2].

Interactive educational systems are in essence pieces of software and therefore any enculturated approach must be expressed in a well-defined, unambiguous manner. In this paper, a preliminary design for building an enculturated interactive educational system namely a web-based Intelligent Tutoring System (ITS) is described together with the features and intended functionality of the various components. Owing to the lack of computationally-viable approaches for building enculturated systems [11], this work contributes a practical method that can be replicated irrespective of the instructional domain chosen. The implementation details and tools used to create two concrete systems based on this architecture for the Computer Science domain are discussed. The results of an experimental study and qualitative evaluation conducted using the systems are outlined. An analysis of the findings and empirical evidence reported in the study is done together with a discussion of the significance of the results. The paper concludes with a summary of the research contributions made and the future plans for improving the prototypes and advancing the research ideas expressed.

II. ENCULTURATED SYSTEM DESIGN

Enculturated instructional software systems must satisfy several requirements, and design decisions were made in order to satisfy these requirements. A modular design was chosen because of the complexity involved in delivering a culturally-relevant instructional experience. Flexible alteration and improvement of the component features are easily accommodated as a result. Many of the components featured in the design of the enculturated web-based ITS are based on the following traditional ITS components: a student model, a domain model, and an expert model [8].

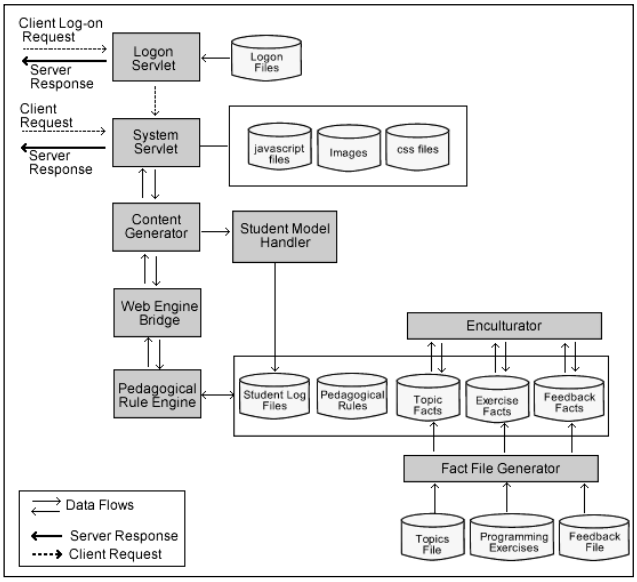

Figure 1. Enculturated web-based ITS system architecture

A client-server approach is taken. A cultural student model, cultural heuristics, a content repository, and a pedagogical module make up the major architectural units of the design. Fig. 1 shows how these components are connected in a cohesive system and it also shows the exchange of data amongst the components. Web related components such as server software and content aggregators will not be discussed since these are standard in web-based systems.

A. Cultural Student Model

A student model serves the traditional purpose of recording the student's knowledge levels, learning achievements, and learning goals. It stores logs of the pedagogical events in the web session. The student model stores performance-related data such as prerequisite knowledge, topics completed, submitted answers, questions completed, successful and failed attempts at learning activities, number of attempts, time taken, suggested hints and instructional guidance given to the student. Economides [3] and Blanchard et al. [1] recommend storing cultural learner-related data in a student model and consequently the cultural learner model was assimilated into the student model since they are inherently related. The model also tracks the player's interaction with the software, and records information related to how the system is being used such as dwell time on areas of the screen for example. It is essentially a snapshot of the player's educational experiences.

B. Pedagogical Module

The pedagogical module serves the same purpose as the expert model in an ITS, but was separated from the student model for a cleaner design. Instructional rules constantly access and update the student model in response to input data from the student and events captured from the screen. They control how and when instructional feedback is given, and they manage the selection and transition of the learning activities. Any on-screen feedback given is stored in the student model so that records of student experiences are kept up to date. In order for these rules to properly scaffold the learning activities and determine appropriate feedback for the student, the learning materials need to follow a template used internally or provide metadata descriptions that identify the corresponding parts identified in Table 1 below.

TABLE 1. LEARNING MATERIAL TEMPLATE USED BY THE PEDAGOGICAL MODULE AND CULTURAL HEURISTICS*

Name of Variable	Description of Learning Activity Variable
ID	Unique identifier
Topics_Tested	Topic(s) tested or covered in the learning activity
Multimedia*	Filename(s) of multimedia content used in the activity
Skill_Level	The skill level that a student must possess for successful completion of the activity
Learning_Objectives	Skills or knowledge that a student should possess upon successful completion of the activity
Question_Description*	Descriptions, instructions, and/or scenarios that guide the student and set the context of the question
Question	Content that the student must manipulate, modify, or select from.
Answer	Model answer/content expected from the student
Hints*	Further guidance corresponding to different parts of the model answer

C. Cultural Heurisitcs

A culturally-relevant instructional approach requires that cultural references made by the software system should be applicable to the learning content, familiar to the students, authentically rendered, and integrated into the context of the instructional material [4, 6]. Cultural rules modify the textual portions of instructional content such as question descriptions, scenarios, hints, and instructional feedback produced by the web-based system. The enculturation of visual portions of these systems, namely the multimedia related to the learning activities, is also handled by these rules. The textual modifications include customizing the language of the instructional feedback, whereas the multimedia enculturation involves swapping in cultural assets for generalized assets such as images. Textual outputs of the enculturation process are sentences expressed in a cultural dialect specifically mesolect forms[1], and equivalent cultural lexical terms. The cultural target is determined by the student's cultural background (from the student model).

[1] A variety of language in a Creole continuum that is intermediate between the standard form (acrolect) and forms that diverge greatly from the standard form (basilect).

D. *Content Repository*

The content repository handles the organisation and distribution of all web-related, instructional, and cultural assets to the web content aggregator. Enculturated web-based ITSs rely on reusable content more than non-cultural ITSs because of the additional dimension of cultural personalisation; this was the basis for having a separate asset repository - reusability. The content repository primarily hosts all of the educational and interface-related material used by the system and the student model. For example, multimedia files related to the interface's look and feel, such as icons, logos, and those related to the learning exercises (scenario pictures, feedback pictures) are stored here together with educational material such as question descriptions, solutions, feedback files, topic hierarchies. Each of these assets is described by their asset metadata. The metadata descriptions define the context of use and the nature of the assets and are indispensable in the design because they facilitate reuse and exchange of compatible assets. Both the pedagogical module and cultural heuristic component use these descriptions when making instructional and enculturation-related decisions.

III. SYSTEM IMPLEMENTATION

Two web-based ITSs were implemented based on the software architecture described in the previous section. One system was enculturated for a Trinidad and Tobago context (Culturally Relevant Instructional Programming System – CRIPSY) while the other remained generic (Instructional Non-CulturAl Programming System – INCAPS). Both systems were built for the same educational domain, Computer Science programming, and were identical from a functional standpoint.

The systems were implemented using Java-based tools and technology which facilitated seamless integration of the various components into one complete system. At the presentation layer, HTML, javascript, and css were used to create the web pages and graphical user interfaces. The Dynamic Web Content Aggregator was implemented using servlets that rendered and formatted the web pages. Additional servlets were used to deliver required functionality such as validation, verification of input data, and for handling log-ons and session management. Apache Tomcat 6.0 was used as the web server environment. The pedagogical module, student model, and cultural heuristic component were implemented using JESS (Java Expert System Shell) rule engines and rules. Intermediary java programs were used to connect each of these self-contained units. At the data level, simple file formats were used to manage the content repository since rule engines handle data manipulation primarily using facts.

The development of enculturated assets for CRIPSY was done semi-automatically and manually. Computer Science (CS) education stresses the importance of analytical programming skills [5] because a large part of the curriculum involves reading, planning, and writing computer programs. These skills include being able to understand code written by other people such as libraries and full programs, and being able to detect and repair errors in the syntax and logic of program code. Proper development of these skills requires rigorous practice sessions with written problem sets in the form of code. Basic skills of understanding program code and detecting and repairing syntactical and logic/analytical errors were targeted using code snippets related to topics on the programming curriculum of the target student audience. Constructivism and situated cognition was selected as the major instructional strategy since analytical programming skills were being targeted and also because of the good fit between situated cognition and culturally-aware instruction

The programming exercises' descriptions, parts of the exercise code and instructional hints were enculturated using subtle, careful use of cultural semiotics, specifically familiar language and cultural names of objects and foods.

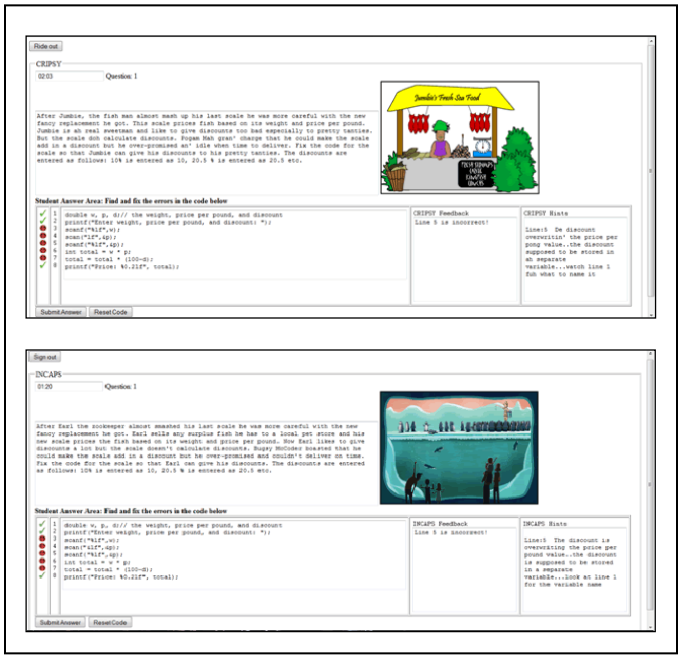

Figure 2. Screenshots of CRIPSY (top) and INCAPS (bottom) featuring enculturated and non-cultural versions of the same programming exercise.

As shown in the screenshots in Figure 2 above, both systems used the same instructional content but differed in the expression of the content, that is, cultural and non-cultural. A minimalist interface design was used in order not to distract the students and to increase ease of use. Instructional feedback consisted of identification of correct/incorrect lines of code, hints for the incorrect lines, and general guidance.

IV. EXPERIMENTAL STUDY AND RESULTS

A study was done to evaluate whether the enculturated system, CRIPSY, was more effective than the control system, INCAPS, for increasing analytical programming skill, and to gauge student opinion and interest in the culturally-enhanced approach taken.

A. *Participants*

Sixty (60) students, 31 males and 29 females, enrolled in the first year computer programming course at the University of the West Indies (U.W.I.) voluntarily participated in the study. Aged between 18 and 47 years (mean=20.983, s.d.=4.835), 23.3% were of African descent, 41.6% were of East-Indian descent, 1.66% were of Caucasian descent, and 33.3% were of mixed ethnicities.

B. *Procedure*

The students were randomly assigned to a control group (n=30) and a test group (n=30). A timed pre-test was administered to both groups, then the students logged on to

the test servers in the Computer Science laboratory at U.W.I. Each student was given a unique username and password that activated the respective pre-assigned system. The student was therefore unaware of whether the system was cultural or non-cultural prior to logging on. The test group used CRIPSY and the control group used INCAPS. After 30 minutes, the sessions timed-out automatically to ensure that both groups used the systems for the same duration. A timed post-test was then administered, followed by an evaluation survey. At the end of the experiment, usage logs were retrieved from the server machines.

C. Results

Initial examination of the pre-test (P1) and post-test (P2) scores indicated positive changes in the post-test performance of both groups of students as illustrated in Table 2 below. Paired t-tests conducted on both sets of pre-test and post-test scores revealed statistically significant increases in the students' programming skills after they used the systems. Table 2 shows that the difference in the post-test scores of the test group (CRIPSY) was higher than that of the control group (INCAPS). It should be noted however, the difference between the groups' scores is modest and not statistically significant (p= 0.3975)

TABLE 2. CHANGES IN PRE-TEST AND POST-TEST SCORES FOR THE TEST GROUP (CRIPSY) AND CONTROL GROUP (INCAPS)

Group	N	P1	P2	Diff	T-Test
CRIPSY	30	8.0667	10.1000	2.0333	0.0001
INCAPS	30	8.4333	9.9000	1.4666	0.0045

Overall, the students had roughly equivalent averages for time-on-task with the programming exercises for the test and control groups. Greater variation was found for the number of correct, incorrect and total attempts made at completing the exercises between the test and control groups as shown in Table 3 below.

TABLE 3. SUMMARY STATISTICS EXTRACTED FROM SESSION LOG FILES FOR THE TEST GROUP (CRIPSY) AND CONTROL GROUP (INCAPS)

	CRIPSY	INCAPS
Correct Questions	m = 1.8077 s.d.= 1.4148	m = 2.2174 s.d.=1.9059
Correct Attempts	m = 9.0385 s.d.= 5.2267	m = 11.2174 s.d.= 7.7222
Incorrect Attempts	m = 26.2692 s.d.= 12.3792	m = 37.5217 s.d.= 18.3424
Total Attempts	m = 35.3077 s.d.= 16.2327	m = 48.73913 s.d.= 22.05204
Time-on-task	m = 23.52745 s.d.= 6.041183	m = 25.58684 s.d.= 6.373995

Analysis of the logged session data revealed positive, very significant linear correlations ($p<0.01$) between the total time spent on the exercises in the systems and the total number of attempts made for both the test ($r=0.584$) and control groups ($r=0.519$). Strong, extremely significant correlations ($p<0.001$) were also found between the total number of correct attempts made and the total number of exercises completed successfully for both the test ($r=0.899$) and control groups ($r=0.932$). The total number of incorrect attempts and the total number of exercises completed successfully were very significantly correlated for the test group ($r=0.519$, $p<0.01$) but only weakly related for the control group ($r=0.102$, $p>0.1$). A strong significant correlation was found between the positivity of the system rating given by the test group students and the total number of attempts they made at the exercises ($r=0.589$, $p< 0.01$) whereas a weak negative correlation existed for the control group ($r=-0.097$, $p>0.1$).

In the subjective assessment survey, 56.2% of the test group students rated CRIPSY as 'really good' and 'pretty good', 42.3% rated it as 'ok', and 11.5% rated it as 'not good'. The most popular reasons for liking CRIPSY were helpful hints, interesting and funny problem contexts, encouraging understanding of programming errors, and the use of cultural language. The most common reasons for disliking CRIPSY included server glitches, lack of flexible answer formats, unreadable or "too much" cultural language, and confusing problem contexts.

43.5% of the control group students rated INCAPS as 'really good' and 'pretty good', and 56.5% rated it as 'ok'. None of the control group students rated INCAPS as 'not good'. The most popular reasons for liking INCAPS were interesting and funny problem contexts, encouraging understanding of programming errors, and helpful problem descriptions. The most common reasons for disliking INCAPS included server glitches, lack of flexible answer formats, unreadable cultural language, and distracting problem contexts.

The difficulty of the exercises received mostly similar reviews. Around 4% described the exercises as challenging from both groups. 74% of the control group and 76% of the test group found the exercises to be neither easy nor difficult. The remaining 22% of the control group rated the exercises as easy compared to 16% of the test group. Interestingly, 4% of test group gave a rating of 'very easy'. The instructional feedback was identical in both systems with the exception of the enculturated language used in CRIPSY. Students rated the feedback in CRIPSY as helpful (85.7%), encouraging (10.7%) and useless (3.6%) whereas INCAPS was rated as helpful (69%), encouraging (24.1%), and confusing (6.9%).

V. ANALYSIS OF RESULTS

The increases in test scores for both groups were significant and provide strong evidence for the success of Intelligent Tutoring Systems in increasing student performance. Although only marginally larger, the group that used the enculturated system, CRIPSY, produced larger gains compared to control group. This confirms the assumption that cultural interventions do indeed have positive effects on learners [6, 7] and provides empirical evidence in support of enculturated learning systems. In addition, the seamless integration intended amongst the cultural software components and the traditional ITS components was achieved since similar usage patterns were found in both the test and control group session logs. Cultural elements did not detract away from student behaviour commonly reported for ITSs since similar correlations were found for between the total time spent on the exercises in the systems and the total number of attempts made, and between the total number of correct attempts made and the total number of exercises completed successfully for both experimental groups. The enculturated system however performed slightly better compared to the control system since the total number of incorrect attempts and the total number of exercises completed successfully

were strongly correlated. This implies that both correct and incorrect attempts made with the enculturated system proved to be beneficial for completing the exercises. Making attempts also seemed to be encouraging for students in the test group since the positivity of their system ratings increased with the number of attempts made. This relationship was weaker for the control group which suggests that the culturally enhanced system may indeed have promoted a more relaxed learning atmosphere as theorized by proponents of culturally-aware instructional environments [6, 7].

Both systems suffered from parallel problems related to server glitches and software bugs. Inflexible answer formats understandably caused confusion for both sets of students since correct alternative programming solutions were not accepted by the pedagogical module. This resulted in some students being blocked from advancing to the next exercise despite having correct answers. Surprisingly, these problems did not decrease the appeal of the tutoring systems because students were eager to have access to the systems when the experiment was over. Another design issue that affected students was the static cultural language used in the enculturated system. For some students the density of the cultural terms used distorted the text and made the sentences difficult to read quickly and therefore difficult to understand quickly. An initial suggestion of incorporating a customizable language density scale was proposed as way of dealing with this problem so that users may adjust the language to suit their own preferences. This feature was immediately confirmed as desirable by the students.

Overall, the students liked the enculturated system primarily because of the reasons outlined in earlier studies [9, 10] enriching learning experiences and humour. The use of culture created a familiar setting and it was done in a way that was interesting to the students. Although both systems employed an anchored instructional approach, a larger percentage of students rated the programming exercises as easier for the enculturated system compared to the control system. The instructional feedback/hints were also deemed to be more helpful by the students using the enculturated system again despite providing similar guidance to those used in the control system.

VI. CONCLUSION AND FUTURE WORK

Culture is rapidly becoming an important consideration in the design of eLearning software firstly because of the increase in the number of users accessing software over the Internet, and secondly because of the sheer diversity in the cultural backgrounds of these users. Conventional learning has often taken place in a localized setting with a teacher guiding one or more students in their search for knowledge. With the advent of the Internet, this traditional setting has changed drastically since students now have access to teachers and educational material from over wide distances. Consequently, these students are exposed to a variety of educational tools, teaching strategies and learning materials which were not developed with their own personal needs in mind. This has dramatic usability implications especially when the mainstream culture for which e-Learning materials are designed clashes with that of the users.

Based on the encouraging evidence established by the study, the research discussed in this paper demonstrates a practical approach towards developing an enculturated web-based learning environment. By leveraging research from various fields such as Intelligent Tutoring Systems and culturally-aware instruction, this work shows how the complexity of enculturation can be managed and how aspects of intelligent tutoring can be enculturated. Empirical evidence indicates that enculturated systems perform as well as traditional tutoring systems and are potentially superior at creating relaxed, engaging learning atmospheres for the Computer Science programming domain. However care must be taken to ensure that the cultural enhancements match the tolerance level of the student users. Further refinement and improvements are planned for the systems described. A limited amount of cultural automation was undertaken, so expansion of the cultural coverage is necessary. Additional features such as deeper cultural learner profiling, adjustable language density, and greater tutoring flexibility are also part of the plans intended for this research.

REFERENCES

[1] E. Blanchard, R. Razaki, and C. Frasson, "Cross-cultural adaptation of e-Learning contents: A methodology," In: Richards, G. (ed.): Proc. ELearn 2005, AACE, 2005, pp. 1895-1902.

[2] E.G. Blanchard, and R. Mizoguchi, "Designing culturally-aware tutoring systems: Toward an upper ontology of culture," In E Blanchard, and D Allard (eds.), Proc. Culturally Aware Tutoring Systems, 2008, pp. 23-34.

[3] A.A. Economides, "Culture-aware collaborative learning," Multi-cultural Education & Technology Journal, vol. 3, issue 4, 2008, pp. 243-267.

[4] M. Fleer, "Reflecting indigenous culture in educational software design," Journal of Reading, vol. 32, issue 7, 1989, pp. 611-619.

[5] S. Gray, S. Edwards, G. Lewandowski, and A. Shende, "Improving student programming skills by developing program comprehension abilities: Panel discussion," Journal of Computing Sciences in Colleges, vol. 20, issue 3, 2005, pp. 235 – 237.

[6] L. Henderson, "Theorizing a multiple cultures instructional design model for e-Learning and teaching," In Globalized E-Learning Cultural Challenges, A. Edmundson, Ed. UK: Information Science Publishing, 2007, pp. 130- 154

[7] C. McLoughlin, "Adapting e-Learning across cultural boundaries: A framework for quality learning, pedagogy, and interaction," In Globalized E-Learning Cultural Challenges, A. Edmundson, Ed. UK: Information Science Publishing, 2007, pp. 223- 238.

[8] C. Mills, and B. Dalgarno, "A Conceptual model for game-based Intelligent Tutoring Systems," Proc. Ascilite ,http://www.ascilite.org.au/conferences/singapore07/procs/mills.pdf, 2007

[9] P. Mohammed, and P. Mohan, "Student attitudes towards using culturally-oriented educational games to improve programming proficiency: An exploratory study," Proc. Edutainment, LNCS 5670, Springer, 2009, pp. 196-207.

[10] P. Mohammed, and P. Mohan, "Combining digital games with culture: A novel approach towards boosting student interest and skill development in Computer Science programming," Proc. Second International Conference on Mobile, Hybrid, and On-Line Learning, IEEE Computer Society, 2010, pp. 60-65.

[11] M. Rehm, "Developing enculturated agents- Pitfalls and strategies," In Handbook of Research on Culturally-Aware Information Technology: Perspectivies and Models, E.G. Blanchard, and D. Allard, Eds. Hershey: IGI Global, 2010, pp. 362-386.

[12] P. Young, "Integrating culture in the design of ILTS," British Journal of Educational Technology, vol. 39, issue 1, 2008, pp. 6-17.