AUTOMATING THE PROCESS OF SKILLS-BASED ASSESSMENT

Mike Joy, University of Warwick

INTRODUCTION

Computing and Computer Science students must acquire a variety of skills early on in their undergraduate career, including the ability to write computer programs, and to construct and reason about simple algorithms used in programs. Not only are these fundamental to their academic progression, but they are also practical skills which cannot be mastered by reading books or viewing web pages: students must practice programming. There are many excellent books and web-based resources which facilitate the learning process, but the assessment of programming skills has been an activity requiring substantial human resources. It should be possible, however, to automate the assessment process, either completely or in part, since program code is in a form suitable for automatic processing. In this chapter the pedagogical, technical and practical issues which have affected the deployment of automatic assessment of computer programming skills will be examined. The tools and packages currently (2003) available to assist in the automation of assessment will be discussed.

COMPUTER-ASSISTED ASSESSMENT (CAA)

Before discussing programming-related CAA issues, it is helpful for us to stand aside and consider what is actually meant by "assessment", and how automation of assessment is important in a more general context. Bloom's Taxonomy (Bloom, 1956) provides a framework to start thinking about the purpose and depth of assessment. The Taxonomy classifies six levels of learning objectives:

- Knowledge
- Comprehension
- Application
- Analysis
- Synthesis
- Evaluation

Levels 1-3 are sometimes described as relating to "shallow" or "surface" learning, and levels 4-6 as "deep" learning. Assessment of level 1 is relatively simple, and can frequently be accomplished by Multiple Choice Questions (MCQs), or questions requiring simple responses. It becomes progressively more difficult to measure a student's competence as the higher level objectives are addressed.

In the UK higher education context, there is a loose mapping between the National Qualifications Framework (NQF) six level qualification descriptors (QAA, 2000) and the levels of Bloom's Taxonomy. Each NQF level requires a mixture of competencies, abilities and skills which include all the Bloom levels of cognitive ability. At the lowest level, HE1 (Certificate of Higher Education), the emphasis is on knowledge of a subject and the ability to apply that

knowledge, whereas the description of level HE6 (PhD) concentrates on the creative and evaluative attributes of a higher degree.

The types of simple question which are often used in CAA tools to test surface learning include:

- Multiple Choice Questions (MCQ), with a single choice of response
- Multiple Response Questions (MRQ), similar to MCQs but with multiple selections of responses
- True/false questions
- Hotspot questions, where a point or area on an image is selected
- Text questions, requiring a response which is the input of text
- Numerical questions, requiring a numerical response
- Matching questions, where items in two lists are paired off
- Ranking questions, placing items in an order according to certain criteria

There are variations on these themes, such as questions requiring a mathematical expression as the response, or perhaps a fragment of program code. Where simple knowledge or comprehension is being measured, a combination of these types of question can be very effective. However, creating effective questions is not necessarily a straightforward task. For example, when writing MCQs, it is necessary to ensure that clues to the correct answer are not accidentally included either in the question or in the choice of potential answers. There is now substantial literature defining best-practice when composing such questions, and this is summarised in the CAA Centre's Blueprint (Bull, 2001).

It has been argued (Entwistle, 2001) that such simple question types cannot be used to measure deep learning, and that the higher levels of cognitive ability are best measured by techniques such as essays and problem-based questions. Automatic marking of complex solutions, such as essays, is difficult.

Assessment can be either summative or formative. Summative assessment is simply used to measure students' performance, and is typically an examination under controlled conditions. Formative assessment is additionally used to provide constructive feedback to students and therefore also has a role in the learning process (Black, 1998). Specifically, Wiliam and Black specify that "in order to serve a formative function, an assessment must yield evidence that, with appropriate construct-referenced interpretations, indicates the existence of a gap between actual and desired levels of performance, and suggests actions that are in fact successful in closing the gap" (Wiliam, 1996). A key application of CAA is in formative assessment, and CAA may assist the monitoring of students' progress through frequent testing.

ASSESSMENT OF PROGRAMMING SKILLS

When measuring students' programming skills it needs to be clear exactly what is being assessed. The following are (some) attributes of the process of

writing a computer program which may be measured, in an approximate ascending order of level of cognitive ability required by the student.

- Comments in code
- Code style (layout, choice of identifiers)
- Correctness of code
- Code structure (use of language artefacts)
- Code testing
- Use of external libraries
- Design documentation
- User documentation
- System documentation
- Efficiency of code
- Choice of algorithm
- Efficiency of algorithm used

Of course, not all of these skills will be acquired at the same time, and a student's programming skills will continually be developed during a computing degree course. Some of these attributes can potentially be measured (such as correctness of code), whereas assessing documentation is not dissimilar from assessing an essay, albeit a short and formalised one. There is an element of subjectivity, especially regarding style and documentation, and programming is often regarded as an art.

How can Bloom's Taxonomy be applied to these skills? The knowledge base required for a programmer is relatively small. Basic programming is synthetic - it is a creative skill - and with only a surface understanding of computer language concepts it is not possible to be a competent programmer. Most of the attributes listed above require at least the application of a student's knowledge of a programming language, and are at the middle to upper levels of the Taxonomy. Programming is a skill which requires deep learning, and, as remarked above, the simple dialogues available in generic CAA tools are arguably not appropriate for measuring such a skill.

ISSUES

Before using any CAA package, there are a number of factors which an academic should be aware of. Many of these are security-related, and of a technical nature, but all of them impact on the use or choice of a CAA tool. Some are potentially crucial if a tool is used for summative assessment, but may be unimportant for a minor formative exercise or for student self-testing.

Authentication of the student user of a CAA tool may be weak. It is usually necessary for the teacher to ensure that the student attempting a piece of work is the student they claim to be. It is easy for students to tell each other user codes and passwords, and for a student to impersonate another, and this is problematic when an assessment is not held under controlled examination conditions.

Does the tool interface with the institution's student record software?

Access to accurate student data potentially improves the ease with which a module can be managed.

Are local passwords/IDs used? Allocation and management of passwords must be performed safely, for example preventing the use of guessable passwords (Frisch, 1995).

How is sensitive information (such as passwords) stored? It must be practically infeasible for another user of the system to access that information, which should be stored encrypted. Is a student disconnected after a period of inactivity? Students can accidentally leave themselves "logged on", and another person could then continue the session. Most CAA tools are client-server software, and this raises network-related problems.

If a specific platform is required for a client, do all students have access to suitable equipment? Although some vendors assume all PCs run Windows, in University environments both Linux and Macintoshes are increasingly popular.
Is the connection between client and server encrypted to prevent passwords and other sensitive data being intercepted? "Packet sniffing" and other hacking software can be simple to use, and if the connection passes through an insecure network unencrypted data can be read (McClure, 2001).
What happens to a session when the client-server connection is unexpectedly broken? If a student is part-way through an assessment, it is probably undesirable for any data entered by the student to be lost.
Has the tool ensured that documents presented to the browser do not contain data which, when the source is examined, suggest the answers to the questions? Some tools, for example, encode the answers to questions in the URL presented to the browser.

Can a student use the client's features to navigate through the displayed dialogue in an unexpected way? For example, does pressing the "back" button of a browser allow a student to attempt a question multiple times?
Will a browser client work with all up-to-date web browsers? Some web sites are coded so that they are readable with only one browser (typically Internet Explorer), and when viewed with a different browser do not display correctly. The same is true with some CAA tools which use web browsers as clients. In the UK, the provisions of the Data Protection Act 1998 (HMSO, 1998) must be observed. Students have a right of access to data held on them, and any such data must be stored securely. Since the Act enacts a European Union directive, any CAA tool used within the European Union must comply with the corresponding law in the country of use. This is a particular concern if a tool originates from outside Europe, and especially if its origin is the United States, where the emphasis on data protection legislation is different.

Do students have access to data stored on them if requested (subject to the provisions of the Act)?

Is student data stored securely? Data must not accidentally be divulged to unauthorised persons.

Is access to student data restricted to authorised persons?
If CAA is used, and is not under controlled conditions (such as exercises to be performed "at home"), then there is the possibility that plagiarism will take place, and it is important that this is recognised when designing the course. This is dealt with in detail elsewhere in this book, but two issues are appropriate to mention here.

Are students made aware of Institution policy about plagiarism when using the software?

Does the tool contain software to detect plagiarism? If it does not, it may be difficult to know if plagiarism has occurred.

GENERIC PRODUCTS

There is a variety of CAA packages available for purchase, and each is sold as software which performs a wider course management function. Some of these, such as WebCT (WebCT, 2002), form a Virtual Learning Environment (VLE) and have been adopted by some institutions as a major vehicle for course delivery. Few are focussed on assessment, the major exception being Question Mark Perception (QuestionMark, 2002), which is currently an industry standard. All offer a choice of simple question types discussed earlier, together with suitable authoring tools. Most offer a Web-based student interface, although the authoring interfaces may require a WindowsTMplatform. Most are sold as application programs, but some such as WebMCQ (WebMCQ, 2002) are packaged as services with the servers located in company offices. None has any specialised support for assessment of programming skills. Costs vary, but may exceed £10,000 for a site licence. Some products, including Question Mark Perception, offer interoperability with academic student record systems. Some packages have been developed by Universities and by individual academics. These are generally free, or the cost is nominal, but support is seldom offered. Some are part of on-going research, such as the Netquest project (ILRT, 2002) exploring the creation of searchable question banks for online delivery of tutorials and assessment. A danger with non-commercial products is that their development may unexpectedly be halted, such as the TACO project (Sasse 1998).
Funding for products such as Leicester University's CASTLE Toolkit (ULCC, 2002) should ensure their continued availability. As new technologies emerge, software is often not updated to take advantage of them. Some products require the use of third-party software, such as the TRIADS project (Mackenzie, 1999).

A common problem with academic software is that it is not completely packaged, requiring time and expertise to install and maintain. Security is a concern with all these products. None appears to be entirely secure, and not all vendors were prepared to discuss security issues which were raised with them. For products which were intended for purposes of self-testing only, this is not a problem, but most products do claim they can be used for diagnostic or summative purposes. An interesting approach by Question Mark

Perception is the provision of a secure browser which it claims overcomes most security problems.

A major issue is interoperability between products. It may be inadvisable to commit to a specific product which cannot share data with other similar products. Software written for other purposes (such as student record software, graphical design tools, or plagiarism detection software) should ideally work seamlessly with any chosen CAA product. Unfortunately, this does not always happen, but specifications for data storage and representation are being developed. The IMS Consortium (IMS, 2002) is developing XML based standards for the storage of CAA questions, and there are academic initiatives such as the Coresoft project (UW, 2002) defining a database schema and APIs for interoperation with student record systems, and the TML markup (ILRT, 2002) used by Netquest.

Having identified a CAA product, questions and answers must be written before it is deployed. This is a time-consuming task. Is it possible to re-use previously written questions from available question banks? Some of the products reviewed offer learning materials, either as part of the package, or as extras, however their content and/or presentation may not easily integrate with established courses. It is interesting when speaking to academics how many are enthusiastic about question banks, and encourage their development, but are reluctant to contribute material themselves. The issue of intellectual property rights is one which constrains the development of question banks.

There are currently only two products which are aimed at the assessment of programming skills, and which have been deployed outside of a single institution. The remainder of this chapter will examine each in detail.

CASE STUDY 1: COURSEMARKER

CourseMarker (LTRG, 2002), formerly known as Ceilidh and as CourseMaster, has been developed at the University of Nottingham, and is a client-server based system for delivering courses based around programming or diagramming exercises. It allows students to develop and submit programs online, and for those programs to be marked automatically. It is available for purchase, although the cost is relatively low compared with the commercial products discussed earlier.

Security is well-addressed, traffic being encrypted, and cross-checking of session keys employed as a further protection against packet-sniffing. CourseMarker contains plagiarism detection software which will compare students' submissions pairwise and indicate instances of significant similarity.

The software is straightforward to use, both for teacher and student, although installation requires technical expertise.

CourseMarker marks programs via three subsystems. A typographic tool uses predefined typographic properties (metrics such as proportion of comments,

indentation, brace style), and assigns marks dependent on the metrics being within certain bounds. A feature tool assigns credit if a submission contains a given feature (such as use of a specified language construct). A dynamic testing tool captures the output of the program (considered as one or more strings) and compares against expected output.

The main client and server software which forms the majority of CourseMarker are Java applications using RMI. The three marking subsystems also contain scripts in other languages, and can be configured as required (although this may require specific skills, such as competence with regular expressions).

The process of writing and submitting a program allows a student to run the marking tools themselves prior to submission, and to view the marks which would be awarded. Students are thus encouraged to conform to a particular coding convention.

CASE STUDY 2: BOSS

The BOSS Online Submission System (UWDCS, 2002a) has been developed at the University of Warwick. It shares some features with CourseMarker, being a client-server based system, coded in Java, and using RMI. Unlike CourseMarker, no supplementary scripts are used, the code is 100% Java, and is platform-independent having been tested on Sun Solaris, Linux and Windows 95/98/NT/2000. A small bootstrap program is installed on the client machine, which downloads the client software from a central server, thus simplifying the process of upgrading the software.

Security has been a high priority in the software development, and communication between client and server can be SSL-armoured. Access to the software is password protected, and users have access to data on a need-to-know basis. The server stores information in a database which may be populated with student record data is a simple (and institution-independent) way, thus accurate mark sheets can be constructed. BOSS is distributed with plagiarism detection software (Joy, 1999; UWDCS, 2002b)

BOSS automatically marks programs via a single subsystem, similar to the dynamic testing subsystem of CourseMarker. This allows a submitted program to be run, and for a given input its output is compared against expected output. The input and output can either be specified as strings (as CourseMarker) or as instances of Java classes. Submitted programs can either be automatically marked (in which case only dynamic testing is used), or manually (where staff use online forms), or a mixture of both. Feedback is provided to students through the email return of marks, (optionally) broken down by category, and (if appropriate) messages written by staff during the manual marking phase. Thus if there is dynamic testing, students can see which tests failed, and the program output generated, which is information which will help them understand how to correct the program.
An interface to student record information is provided through the use of the

Coresoft database (UW, 2002). There is currently no typographic tool in BOSS.

Staff can make available to students some or all of the dynamic tests, so that students can receive instant feedback on whether their programs are working prior to submitting their programs for assessment.

The philosophy underlying BOSS is that it should be targetted at a single functionality, and as such concentrates on the processes of submission, dynamic testing, and marking. As such, some of the features of CourseMarker are absent, and may remain so.

A major pedagogical concern is that automatic testing of programs may be unsound, and good programs may fail tests for trivial reasons. BOSS is designed to aid the marking process, not to replace it, and even if the only marks for a given assignment are awarded automatically, the marks must be moderated so that programs which fail tests are all marked fairly. In practice, this is not an onerous burden on the teacher, since a well-specified assignment yields few if any such cases.

CONCLUSION

Programming is a skill which requires higher levels of cognitive ability (as defined by Bloom) to master. Generic CAA tools are available which are effective in measuring a student's competency in skills requiring lower levels of cognitive ability, but do not easily target the assessment of a student's programming ability. They are at varying stages of development, with varying functionality, quality, and price, and most lack features specific to the needs of education in Computer Science. There are currently two available tools which have been written to assist in measuring programming skills, which have been described above.

FURTHER INFORMATION

An ongoing exercise for the LTSN Subject Centre for Information and Computer Sciences involves the review of major CAA tools, and a web site (LTSN-ICS, 2002) is maintained. Detailed reviews and comparisons of the individual products are available on the site, together with further CAA resources.

ACKNOWLEDGEMENTS

This chapter summarises work funded by the LTSN Subject Centre for Information and Computer Sciences, and has been assisted by Simon Rawles, Michael Evans and Steven Kumarappan.

Bibliography

Black, P and Wiliam D,(1998) Inside the Black Box: Raising Standards Through Classroom Assessment, *School of Education, Kings College*

London, [Online]
http://www.kcl.ac.uk/depsta/education/publications/blackbox.html

Benjamin S, Bloom and David R Krathwohl (1956) *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*, Longman

Bull, J and McKenna, C (2001) Blueprint for Computer-Assisted Assessment, *CAA Centre, University of Loughborough* [Online]
http://www.caacentre.ac.uk./

Frisch, A (1995) Essential System Administration, pp. 148-154, O'Reilly

HMSO, (1998) *Data Protection Act 1998*, [Online]
http://www.hmso.gov.uk/acts/acts1998/19980029.htm

Entwistle N, (2001) *Promoting Deep Learning through Assessment and Teaching*, American Association for Higher Education, Washington, DC,

IMS Global Learning Consortium, (2002) *Question and Test Interoperability Specification*, [Online] http://www.imsproject.org/

Learning Technology Research Group, (2002) CourseMarker, *University of Nottingham* [Online] http://www.cs.nott.ac.uk/CourseMarker/

LTSN Subject Centre for Information and Computer Sciences, (2002) *Computer-Assisted Assessment*, [Online]
http://www.dcs.warwick.ac.uk/ltsn-ics/resources/caa/

Institute for Learning and Research Technology, (2002) *Netquest*
http://www.ilrt.bris.ac.uk/netquest/

Luck, M and Luck, M (1999) Plagiarism in Programming Assignments, *IEEE Transactions on Education,* Vol 42 No 2, pp 129-133

Luck M and Joy, M (1999) A Secure On-line Submission System, Software - *Practice and Experience* Vol 29 No 8, pp 721-740

Don Mackenzie, (1999) *Recent Developments in the Tripartite Interactive Assessment Delivery System* (TRIADS) [Online]
http://www.derby.ac.uk/ciad/lough99pr.html

McClure, S, Scambray, J and Kurtz, G (2001) *Hacking Exposed (Third Edition),* Osborne/McGraw-Hill

The Quality Assurance Agency for Higher Education, (2000) *The National Qualifications Framework for Higher Education Qualifications in England, Wales and Northern Ireland: a Position Paper* [Online]
http://www.qaa.ac.uk/crntwork/nqf/pospaper/contents.htm

Question Mark Perception, (2002) *Question Mark Perception* [Online] http://www.questionmark.com./uk/home.htm

Sasse, MA, Harris, C Ismail I and Monthienvichienchai, P (1998) Support for Authoring and Managing Web-based Coursework: The TACO Project, in *The Digital University: Reinventing the Academy*, eds R. Hezami, S. Hailes and S. Wilbur, Springer-Verlag, University of Leicester Computer Centre, (2002) *CASTLE Toolkit*, [Online] http://www.le.ac.uk/castle/

The University of Warwick, (2002) *Coresoft* [Online] http://www.dcs.warwick.ac.uk/coresoft/

The University of Warwick Department of Computer Science, (2002a) *The BOSS Online Submission System*, [Online] http://www.dcs.warwick.ac.uk/boss

The University of Warwick Department of Computer Science, (2002b) *The Sherlock Plagiarism Detection Software*, [Online] http://www.dcs.warwick.ac.uk/sherlock

WebCT (2002) [Online] http://www.webct.com/

WebMCQ (2002) [Online] http://www.webmcq.com/

Wiliam D and Black, P (1996) Meanings and Consequences: a Basis for Distinguishing Formative and Summative Functions of Assessment?, *British Educational Research Journal* Vol 22 No 5, pp. 537-548