Teaching on the Wiki Web

Joseph Bergin
Pace University, New York City: berginf@pace.edu • http://csis.pace.edu/~bergin

A wiki is a completely interactive web site. Any page can be edited by any visitor. It is driven by a specialized web server or set of cgi scripts generating dynamic pages from the results of visitor edits. A visitor edits a page by grabbing the current content of a page in an ordinary web form and editing it arbitrarily and then saving it back. Users can also create new pages. Generally the name of a page is the name of a topic. Most wikis are text only. The syntax of a wiki is simpler than HTML, but HTML is recognized by some wikis. Some wikis are very easy to administer. There is no history or automatic backup, however. A wiki is totally "live."

I have a jwiki (simple java wiki) for each of my courses. Students reach it through a URL with a non-standard port number. It gives me a way to communicate easily and asynchronously with them on course topics. They also use it to communicate with me and each other. I also use it to quickly dispel misconceptions and correct errors I might make in class.

Some features of my course wikis.

Home pages. Each student can put up a home page in which they say some things about themselves.

Anonymous feedback. This is a page of unsigned posts about the course. It is a place to gripe if it is needed and a place where students can respond to the gripes of others.

Search: The wiki supports full text search, giving a list of pages with the search terms.

Grow your own FAQ. On this page students post questions that they or I can answer. The resulting list of answered questions can be used in subsequent courses easily.

Not an assignment. Ideas related to the course (or not) that gives them things to explore. Anyone can add to this of course. It has pointers to auxiliary material, etc.

Discussions. In some courses these pages are used to run threaded discussions on a variety of topics. Topic leaders post the initial questions.

Captured chats. I do online chats in some courses. I capture the results and post them for those that can't attend.

Recent Changes. Any visitor can get a list of pages changed in the last week.

Infrastructure hints. These pages include hints on how to use the course software effectively and to solve problems that arise.

The wiki I use is an open source Java project. The original wiki (http://c2.com/cgi/wiki?WelcomeVisitors) was written in perl by Ward Cunningham and his is the virtual home of the software patterns community. It is also used for collaborative projects of all kinds, from online committee meetings to paper preparation.

Very Active Learning of Network Routing

Lillian N. Cassel
Villanova University; Villanova PA USA; cassel@acm.org

Active learning promises students will really own what they have learned because they have done something rather than only read or listen. The following classroom exercise has been used at both the undergraduate and graduate level Computer Networks courses. There has been no formal study of its effectiveness, but an informal result is telling: Two routing algorithms presented in class. On an exam, students are asked to choose one of them and describe it. In several years of using this classroom exercise, nearly all the students chose the method we did with this exercise and they all explained the algorithm correctly.

The classroom exercise: Distance Vector Routing

A number of pieces of string are required, each with a label on each end. The label identifies a router. The strings indicate connections. Each router has several strings. The students do not see the topology of the network. Each student takes the part of a router. If there are more students than routers, the students can work in pairs with one passing information and the other recording information. Each router starts with an empty routing table that must be filled in. The first entry in the table is the router's own ID. The distance is entered as 0. Next, the router looks to see to what other routers it is connected. It enters the ID of each of its adjacent routers into the table, with a distance of 1. Now, the router exchanges its table with all its neighbors and learns of other routers that are two steps away. For each newly found location, the router enters its ID, the ID of the router through which the new one is reached at a distance of 1 plus the distance from the neighbor. If the same destination is discovered from more than one neighbor, the router chooses the one with the smallest distance. After recording the information obtained, the router again exchanges its table with its neighbors, discovering yet more potential destinations. Soon, the router stops seeing new destinations and knows how to reach every site in the network. The router does not know all the connections, but it does know how to reach every destination.

At that point, each router is given a message to deliver and the messages move through the network, being stamped with the ID of each router it passes. We soon see that all the messages are delivered, and that each follows the shortest available route. And we still don't know how the network is connected.

The classroom gets interesting during this exercise. It is generally not possible for everyone to stay seated. They get up and move around to be able to reach the endpoints of the string. One year a string was tangled and we could not get it to unravel, so the students suggested that we had a wireless connection and happily continued.

The exercise fits easily into one class period and the results appear to stay with the students after we are finished.

ITICSE'02, June 24-26, 2002, Aarhus, Denmark.
ACM 1-58113-499-1/02/0006.

A Diagnostic Technique for Addressing Group Performance in Capstone Projects

Tony Clear

School of Information Technology, Auckland University of Technology • Email Address: Tony.Clear@aut.ac

INTRODUCTION

“Students’ awareness of their own group processes can be poor (or Absent)” [1]. Yet professional software development situates much work within groups, and capstone courses are often designed to develop capabilities, such as effective teamwork. Supervising group processes can be challenging, and encouraging students to reflect upon group functioning can be a delicate exercise. A technique is outlined below, for group performance review and diagnosis of issues preventing teams functioning effectively.

THE TECHNIQUE

This technique is adapted from that proposed in [1], p. 249 for “moderation using student input” and has been applied using the “peer and self evaluation forms” on p.250 - 251. The technique is expressed below using the pattern provided in [1] for a bundle of transferable practice.

This bundle allows supervisors to facilitate a group reflective process, which supports group improvement and diagnoses areas of group dysfunction.

The way it works is that the supervisor arranges a face-to-face meeting with the group, at which peer and self-evaluation forms are issued to team members. Each student completes a set of forms, one for each member of the team including themselves. These are handed to the supervisor who briefly reviews them to compare significant disparities. The reviews for the first member of the group are then laid out, and ratings compared, and discussed on an item-by-item basis. Significant disparities are noted and feedback is elicited from the rater(s) and the person being rated.

It works better if the facilitator emphasizes the subjective nature of the evaluations, the fact that these are perceptions to which each person is entitled, and the goal is to improve the performance of the team, or diagnose its issues. It is also an opportunity for each member of the team to experience a form of performance review and direct feedback, and to learn about themselves from the process. It also works better if the exercise is conducted in a spirit of openness and generosity, and manages to avoid judgment and defensiveness. At the conclusion of the meeting, the facilitator summarizes the outcomes and thanks the group members for their contribution to the process.

It doesn’t work unless students are willing to take personal risks by exposing themselves to honest criticism, are prepared to be honest and direct about the performance of colleagues, and have some commitment to learning from the process. It makes demands of the facilitator, who can fail to read the situation carefully, manage the criticisms, and resolve the issues that arise in a positive and active manner.

So: use a facilitated meeting for group reflection, performance review and diagnosis.

REFERENCES

Fincher S., Petre M., Clark M., (Eds.) , (2001), Computer Science Project Work Principles and pragmatics Springer-Verlag, London pp. 3 –26.

Improving Feedback from Multiple Choice Tests

William Fone

Staffordshire University, Stoke on Trent, ST4 2AZ, United Kingdom, W.Fone@Staffs.ac.uk

Providing individual feedback to large groups of students can be difficult. By asking students to record their confidence in multiple choice test answers the quality of feedback can be improved. In a “multiple choice question” (MCQ) test the student is asked to make a selection of one or more correct alternatives from a given list in response to a “question stem”.

Students rely heavily upon the teacher to provide academic guidance. To provide individualised guidance to student when group sizes are large can be difficult. Feedback can be used to inspire a student but it may also demoralise. De-motivation may occur if the student does not understand the feedback or can not use it to determine the correct remedial action to improve. To provide advice that a student can use effectively will normally require the advice to be personalised. [1]

The methods used to reduce the distortion that could be caused by guessing include negative marking and value choice [2]. In both these situations the student may pay a penalty for attempting a question where there is an element of uncertainty. A student with doubts may avoid answering some questions if negative marking is being used. This limits the amount of useful information a grader can obtain from looking at an answer sheet.

In an attempt to provide better feedback a system has been devised that seeks to establish the confidence the student has in the given answer. The student is asked to select an answer .

The student is also asked to indicate the level of confidence they have in the selection. Choosing; very confident/confident/ uncertain/ very uncertain.

If a selection is marked very confident and the selection is correct little needs to be said, this is also true if a wrong selection is indicated to be an uncertainty. However if an incorrect selection is flagged as confident or very confident then a misconception has been identified. This provides a filter for the important issues that need to be raised in feedback. Reducing the amount of feedback yet making it more potent. Where low confidence was indicated against correct answers reassuring feedback can be offered to raise confidence and provide stimulation.

CONCLUSION

By asking students to supply an indication of how confident they feel about their answers in MCQs misconceptions and insecurities can be identified. This allows personalised feedback to be given that is concise yet of high value. The overheads to the student are minimal.

REFERENCES

[1] Gipps, C. V., 1994. *Beyond testing towards a theory of educational assessment.* London. Falmer Press.

[2] Walker, D. M., 2001. A note on multiple choice exams with respect to students risk preference and confidence. *Assessment & Evaluation in Higher Education.* Vol. 26 No 3 pp. 261-2

ITICSE’02, June 24-26, 2002, Aarhus, Denmark.
ACM 1-58113-499-1/02/0006.

The Student Record Book: Showing the Value of Documentation

Robyn Gibson
School of Computing,University of Tasmania, Australia,R.Gibson@utas.edu.au

Students in an introductory programming subject are encouraged to keep a record book of their activities. By the end of semester many have come to realise, almost to their surprise, that documentation is useful. The encouragement includes the use of "carrots" and "sticks".

WHAT IS IN THE RECORD BOOK?

It is a folder in which students record what they do and what they find during their study of the subject.

A few components are prescribed. These are:

- Time log sheets. Students are expected to log all the time they spend on work in the subject and to classify each entry into one of four categories: (C)lass, (L)ab, (O)ut of lab, (R)ecording.
- Weekly summaries of the time logs and a self-assessment of current progress and understanding.
- A front page with the student's name and other details.
- The rest of the book contains whatever material students choose.

They are encouraged to include such things as:

- Their findings while working on tutorial activities
- Notes that they record during lectures
- Annotated hard copies of programs that they have written

THE CARROTS AND STICKS

The record book is the only paper that students are permitted to take into the final examination. This is the big carrot.

A few marks are given during semester for having weekly summaries up to date. This is the little carrot.

Students are required to have their record book with them at tutorials. Otherwise they are recorded as absent. This is the stick.

DOES THE RECORD BOOK SUPPORT STUDENTS' LEARNING?

It seems so - in a number of ways.

Keeping a record book means that students are obliged to have all their material for the subject in the same place and organised in some way. This is a skill that many beginning university students lack.

Students are encouraged to use their weekly summaries to reflect on the effectiveness of their study techniques. Tutors, when checking the books, try to make constructive suggestions. For instance, if a student is spending long hours in the lab and not much time out of the lab, the tutor might suggest that they spend more time out of the lab planning and desk checking their programs.

DO STUDENTS REALISE THE IMPORTANCE OF DOCUMENTATION?

It seems so. The value of having everything organised and in one place becomes apparent quickly. Most students work willingly on their books since they realise that this work in the semester will lead directly to an easier time in the exam.

In the longer term, students have seen that documentation allows them to reflect on their work and to see ways to improve.

A Musical Approach to Teaching Design Patterns

John Hamer
Department of Computer Science, University of Auckland. J.Hamer@cs.auckland.ac.nz

In case you missed the signs, design patterns [2] have suddenly become a core programming topic. The Computing Curricula 2001 [1] includes the topic as a major part of SE1 "software design," and in a recent survey [4] software practitioners ranked "software design and patterns" as one of the areas of knowledge they consider most important to their work.

Teaching design patterns, however, is not so easy. Patterns need to be studied in the context of a software design project; to do otherwise renders the topic asomatous. We have met with some success in using "musical composition" to teach a number of standard design patterns, including Composite, Decorator, Visitor, and Factory. The idea of musical composition comes from Paul Hudak's Haskore system [3].

The project centers on the design of a "musical object," by which we mean either a note (with a pitch and duration), a rest (having just a duration), or a combination of musical objects. The combinations can be contemporaneous (as for a chord), or sequential (like a scale). The two primitive music types together with the combinators come together using the Composite inheritance pattern. Decorators can then be added to describe tempo changes (the scaling of the duration of all parts of a musical object), pitch transpositions, and to assign different instruments to parts of a composition.

With the descriptive part of the system so formed, a Visitor can be used to traverse the musical tree and generate a linear sequence of "musical events," which can then be written as type 1 MIDI files for playback on most standard PC equipment. As well as learning about design patterns, students are given the opportunity to try their hand at transcription or composition. Some examples of their work can be found at http://www.cs.auckland.ac.nz/courses/ compsci360fc/ archive/1999/Ass2/PickOfTheBest/.

REFERENCES

[1] ACM and IEEE Computer Society, *Final report of the Computing Curricula 2001 project* (CC2001), <http://www.computer.org/education/cc2001/>

[2] Gamma, E., Helm, R., Johnson,R., and Vlissides,J. *Design Patterns Elements of Reusable Object-Oriented Software*, Addison-Wesley (1994.)

[3] Hudak, P. *The Haskell School of Expression: Learning Functional Programming Through Multimedia*, Cambridge University Press (2000)

[4] Lethbridge, T. "What knowledge is important to a software professional?" *IEEE Computer* (May 2000)

ITICSE'02, June 24-26, 2002, Aarhus, Denmark.
ACM 1-58113-499-1/02/0006.

Self-Assessment as a Powerful Learning Experience

Tami Lapidot
Department of Education in Science and Technology, Technion – Israel Institute of Technology, Haifa 32000 ISRAEL,
lapidot@tx.technion.ac.il

Every teacher would like to have students that are motivated towards autonomous learning with self-enthusiasm. This Tip presentation will offer one method for achieving such a goal.

For three consequential years (1998-2000) I was teaching a "computing literacy teaching methods" course for CSE students in the Technion. The focus of the course was on computing teaching methods and learning processes.

A major part of the course was devoted to a project the students had to develop. They had to collect data, analyze it, organize and represent it to their colleagues. They had to work in small teams and could choose their own topic as long as they were using different computing tools such as Internet, email, spreadsheet, and others.

The projects ranged topics such as: Wine, women in Islam, Michelangelo, UFO, wedding traditions, Greek mythology, and Marathon history.

At the beginning of the course, the students were told the assessment of their projects would be determined together. Several meetings of the course were devoted to discussions and decisions on the assessment and grading of these projects.

In two of the three courses the students developed a complicated assessment method that was a combination of self-assessment, colleagues and teacher assessment. They agreed on 10-13 different criteria for judging their projects, such as clarity, design, creativity, and teamwork. They decided which percentage of the grade should be given to each stage of the data collection, analyzing, organization and representation.

In the third course, students decided to give everyone the highest grade possible 100. We had a non-signed contract that I will accept their decision and give everyone 100 as a final grade in the project (which was 50% of the course grade) and they will do their best to justify this grade. From this point on they had to determine how hard they were willing to work.

As it turned out, I never saw students more committed to their work. They spent more time with their projects than I ever saw before. Their projects were the best one could expect and they became experts in their fields as one could expect from a learning period of half a semester.

These exciting three experiences pose several questions, such as: How far could we go in order to allow our students experience such meaningful learning experiences? Is it possible to determine similar requirements for a project that will focus on hard-core topics of computer science?

ItiCse participants are welcome to share their thoughts about these questions and others that might rise from this Tip presentation.

Evaluating Student Team Project Experiences

Cary Laxer
Rose-Hulman Institute of Technology, Terre Haute, Indiana, USA, laxer@rose-hulman.edu

The first two courses in the computer science major at Rose-Hulman (Algorithm & Program Design and Data Structures) each have a five-week team-programming project as a component of the course. At the end of the projects, in addition to their program code, each team has to submit a user's manual and a technical manual for their product, and give a 15-20 minute oral presentation. The students are also required to evaluate the project experience on an individual basis.

To facilitate the project evaluation, I designed three forms. One form is given to each team leader to complete, one form is given to the other team members, and the third form is used by me to evaluate (i.e., grade) the project. The general team member evaluation form asks the student if, in the student's opinion, everyone contributed his or her fair share to the project, how they viewed their time commitment to the project, what the strengths and weaknesses of their team leader were, to summarize the team dynamics on the project, and to comment on how worthwhile the project was. The evaluation form the team leaders complete asks them about the contributions of their team members; recommendations for "pay raises," "promotions," and "dismissals;" team dynamics; their strengths and weaknesses as team leaders; and how worthwhile the project was. The form I use to evaluate the projects has areas for me to comment on their oral presentation, the software quality, the program design, and the documentation, as well as provide a numeric grade in each area.

I ask all students to sign the forms, making them stand behind what they write, just like I will sign their project evaluation forms and stand behind the grade I give them and the comments I write. I assure them the information they provide is strictly confidential; only I will read the forms. The information is used to adjust borderline grades in the course at the end of the term.

Overall, I have been extremely pleased with the results of these evaluations. I have used these forms for several years and have always gotten well written, almost professional quality comments from my students. They have had plenty of praise for their peers, but they have also told me, in a proper way, when things did not go quite so well. I continue to use these forms when I have team projects to evaluate. Sample copies of the forms will be available at the conference Tips & Techniques session.

CATEGORIES & SUBJECT DESCRIPTORS

K.3.2 [Computers and Education]: Computer and Information Science Education---Computer science education; K.6.1 [Management of Computing and Information Systems]: Project and People Management---Staffing.

GENERAL TERMS: Management.

ITICSE'02, June 24-26, 2002, Aarhus, Denmark.
ACM 1-58113-499-1/02/0006.