Behavioural Issues in Software Development: The Evolution of a New Course Dealing with the Psychology of Computer Programming

John Lenarcic
RMIT University, Melbourne, Victoria, Australia

John.Lenarcic@rmit.edu.au

Abstract

A historical account is presented of the evolution of a new postgraduate level university course dealing with the psychology of computer programming. An outline of the course is presented and its objectives are discussed, as well as possible reasons for its premature demise and initiatives that could be undertaken to resurrect the format in a new guise.

Keywords: Computer Programming, Cognitive Aspects, Education

Introduction

Programming is a highly demanding, meta-level task that involves the design of instructions to direct a computational device in performing functions that amplify or extend human potential in a wide variety of intellectual activities. An understanding of the cognitive underpinnings of computer programming as a human skill would have much to offer in augmenting our fundamental knowledge of general problem solving. Such awareness would also be of benefit in designing human-computer interfaces, programming languages and other software systems that are user-friendlier.

For over 30 years researchers have been attempting to unravel the underlying psychological processes of programming with varying degrees of success, yet few academic courses currently exist that specifically focus on a critical examination of these issues. (Incidentally, the 1998 version of the ACM Computing Reviews Classification Systems has rendered the keyword category "software psychology" as being obsolete.) The philosopher George Santayana once famously quipped that those who do not remember history are condemned to repeat it. Information Technology is a discipline guilty of living always in the present with one eye permanently fixed on the future, yet at the same time oblivious to the lessons of the past. This paper will offer a retrospective outline of the trials and tribulations faced in designing the curriculum for a short-lived new postgraduate level course dealing with the psychology of computer programming.

Details are of a parochial nature in terms of academic environment because the course was and arguably still is an oddity in terms of traditional university IT curricula. Vat (2000) describes the initial evolution and subsequent teaching experience of a junior core course entitled Software Psychology offered in an undergraduate Software Engineering program. However upon reflection the subject as outlined in the latter paper is a reinterpretation of a typical Human Computer Interaction

(HCI) unit, though informed by a greater emphasis on cognitive psychology. The course described within this paper does not shy away from HCI issues as well but instead confronts them via the context of treating fluency in a programming language as being akin to the ultimate in user interface "technology".

The course of study in question was entitled "Behavioural Issues in Software Development" (BISD) - initially offered for the first and last time in Semester 2, 1993 - and was undertaken by postgraduate students as a coursework offering at Masters and Graduate Diploma level within the Department of Software Development at Monash University in Melbourne, Australia. Honours students (i.e. fourth-year level Bachelor's degree) also had the option of undertaking the subject. The course provided students with the opportunity to examine the theoretical and practical knowledge that has been amassed about the behavioural and organisational aspects of software development at that time. The emphasis was on providing a practical overview of the software engineering process from a psychological perspective. Classic empirical studies of critical issues in software development were analysed to inspire and guide students in designing and executing similar experimental activities in a novel vein.

Historical Overview

The "birth" of the psychological research on computer programming can be traced to the seminal work of Sackman (1970). Focusing on time-sharing and batch-processing, the results and critical interpretations of new sets of experiments are described. Stylistic individuality in programmer behaviour is one topic alluded to in the latter text, which also recommends that an effort be made to establish a "scientific study of man-computer problem solving".

Weinberg (1971) was another milestone in the annals of software psychology, as it's also referred to. The central thrust of this text was a broad discussion of programming as an aspect of human performance, in terms of both an individual activity and a group activity. Ideas were put forward as to the thought processes of professional programmers in their daily duties and how this influenced their occupational behaviour. Weinberg (1982) takes note of a criticism voiced over the title of his classic 1971 treatise: Apparently, it was thought that there was no such thing as the psychology of an activity, but only the psychology of a person or persons. This is clearly incorrect, as the "psychology of an occupation" was an area of study as early as the 19th century. Bryan and Harter (1899) have this to say about the sub-discipline mentioned: "A field of research is offered in the psychology of occupations. The chief engagement of every one is the acquisition or exercise of one or another association of habits, such as constitutes skill in a game, trade, profession, language, science or the like. With a little license one may call all of these occupations. In mastering an occupation, doubtless the whole man is involved, body and mind, sensation and movement, thought, interest, imagination, will - innumerable known and unknown aspects of our psycho-physical life." Roe (1977) is a more recent study of the psychology of occupations.

Weinberg's early efforts to encourage self-examination of a programmer's relationship to a computer, in an attempt to increase productivity and improve software quality, indirectly lead to an overall research initiative by computing professionals to improve user interfaces from a human factors perspective. Shneiderman (1980) highlights the achievements of the first decade of this human factors research thrust. The book contains numerous end-of-chapter exercises and ideas for student projects, making it an excellent, if somewhat dated, prescribed reference for any software psychology course (i.e., if university computing departments are bold enough to put such a unit on the books!) Shneiderman (1992) is the new incarnation of the latter text, with the present title conspicuously lacking the ideologically unsound term, "psychology". This edition is very much user-interface-oriented in content, featuring details about such items as interaction devices, gestural input and touch screens.

In terms of forums for the exchange of information, a rather informal group known as the Software Psychology Society, based in Washington D.C., has met monthly since 1976 (Shneiderman, 1986). The regularly held "Workshop of Empirical Studies of Programmers" is a small conference devoted to psychological issues of computing. Freeman (1992) reviews the highlights of the fourth workshop, held in New Brunswick, New Jersey, USA in 1991. Interest in this event must have waned in recent years as the 1999 workshop was postponed due to insufficient paper submissions. Kluwer has published the international journal "Empirical Software Engineering" since 1997 with content tending to focus primarily on experimental analysis of program code rather than on the fuzzier human issues involved in the development process.

Hoc Green, Samurcay, and Gilmore (1990) claim to be the first book to survey the field of programming psychology, dealing with a wide variety of issues including mental representations of a program and debugging aspects. Vaske and Grantham (1990), on the other hand, extensively discuss social and organizational issues of human-computer environments, venturing into the anthropology of computing. One interesting observation about contemporary research into empirical studies of programmers is the shift in association from the established field, of psychology to the trendy new discipline of cognitive science, an amalgam of computer science, psychology and linguistics.

The Birth of an Academic Course

Academic courses dealing with human-computer interaction (HCI) are common within university computing departments. For example, Carey (1989) describes a basic HCI course for software engineers at undergraduate level that places an emphasis on the mechanics of user interface design. Baecker (1989) goes so far as to propose a radical scheme in which the subject content of an entire B.Sc. degree programme within a computer science department is devoted to "user-centred system and interface design". Hewett (1987) describes an undergraduate subject entitled "Software Psychology", based on a modified version of the course framework presented in Shneiderman (1980). The course in question focuses on the cognitive psychology of computer programming and the application of cognitive psychology to the design and evaluation of user interfaces, the greatest emphasis being on the latter.

The new BISD course within the Monash University Department of Software Development was inspired by Curtis (1987), the syllabus for a graduate seminar series designed for doctoral students with a software development background at the Department of Management Science and Information Systems, University of Texas. Originally, the BISD course proposal was entitled something like the "Psychology of Software Development". However, it was suggested that any unit with the word "psychology" in the title would not be officially approved if the unit did not originate from a psychology-based discipline itself. "Management of Software Development" was an alternative title selected but this too was rejected. The final subject name (i.e., BISD) was the result of a group decision by the Faculty of Computing and Information Technology Graduate Studies Committee at Monash University.

Course Structure

The format of BISD was a semester-long series of 4 hour seminars, once a week for 13 weeks. Each seminar consisted of an appropriate blend of "chalk-and-talk" lectures, discussion tutorials, video presentations and guest speakers. The discussion sessions revolved around research papers published in journals and conference proceedings. (Class sizes were fortunately small enough to successfully manage these discussions.) Video presentations and guest speakers were used to bolster the background knowledge of students in key areas (e.g. psychological issues, experimental design, etc.) Total assessment in the course was based on participation in class discussions, as

well as a lengthy written report (4000+ words) and corresponding oral presentation in a specific area related to the course content.

The BISD course syllabus consisted of the following topic areas:

Introduction to Problems of Programming-in-the-Large (i.e. in large team environments) ; Human Cognition and Expert-Novice Differences in Software Development; The Structure of Programming Knowledge in Human Information Processing; Acquisition of Computer Programming Skills; Software Design as an

Individual and Team Activity; The Foundation of Empirical Research in the Behavioural Sciences (with an emphasis on Software Development studies); The Comprehensibility of Representational Formats in Software Design; The Structure and Usability of Programming Languages; Programming Style and Software Metrics; Empirical studies of Software Testing and Debugging; The Selection, Appraisal and Motivation of Programmers; Programming Team and Organizational Structures; Behavioural and Empirical Issues in the Future of Software Development.

Recommended references for the course included Boehm (1981), Curtis (1985), Hoc et al. (1990), Lammers (1986), Mayer (1989), Shneiderman (1980), Soloway and Spohrer (1989), Weinberg (1971) and Weinberg (1988).

Course Objectives

The primary aim of the BISD course was to provide students with an examination of theoretical and practical knowledge about the behavioural and organizational aspects of software development, including a practical overview of the software engineering process from a psychological perspective. In the process of teaching these concepts from examples drawn from existing empirical research, it was assumed that students would find it necessary to grapple with the complexities of measurement theory, statistics, data analysis strategies, research methods and experimental design. Consequently, training in research skills was woven into the fabric of the course as needed to criticize the existing body of research at the time.

It was anticipated that by the completion of the subject, a student should be able to understand the composition of an effective programming team (with respect to such aspects as people, practices and development environments). A student should also be able to critically analyze the software development literature dealing with empirical studies, from both a theoretical and methodological perspective. Such a student should also be able to plan and execute rigorous empirical studies of critical issues in software development, and be able to interpret results from empirical studies with the aim of improving the performance of software development in general.

Ultimately, this course was viewed as a stepping-stone to a possible PhD project dealing with psychological issues in software engineering. Taking this approach, the written report for this subject could have been treated as a PhD research proposal.

Conclusion

Historical details of a new postgraduate course dealing with psychological issues in computer programming have been discussed. It was hoped that this course would have encouraged participants to view software engineering as a human activity, as well as a more formal discipline. However, the course was cancelled by the powers-that-be after its first semester with lack of student interest being cited as the reason. Why the tepid interest? Perhaps it was a course ahead of its time in 1993. IT students of that era were becoming more utilitarian in their choice of subject: Practical courses that covered more commercially fashionable software tools or paradigms were teeming with participants whereas more speculative and traditionally academic offerings floun-

dered in popularity. A course with a "humanities" feel was probably seen as being out-of-place in a panorama of technically saturated curricula and not worthy of consideration.

Plans are underway by the author to resurrect the spirit of BISD in a new guise as a postgraduate subject within the School of Business Information Technology at RMIT University. With the benefit of hindsight, the new incarnation of the unit will be comprised of three distinct parts: 1) a component dealing with a general history of software engineering to place the field in a human context and identify its successes and failures; 2) an updated, slightly compressed reworking of the old BISD syllabus; and 3) an introduction to the ethics of software development. In this way, the proposed course will be a more acceptable marriage of the sciences and the humanities with respect to computer programming. Only time will tell if this can make it more appealing to the masses.

References

Baecker, R. (1989). A vision of education in user-centred system and interface design. *SIGCHI Bulletin*, *24* (3), 10-13.

Boehm, B. (1981). *Software engineering economics*. Englewood Cliffs, NJ: Prentice-Hall.

Bryan, W. L. & Harter, N. (1899). Studies on the telegraphic language. The acquisition of a hierarchy of habits. *The Psychological Review*, *6* (4), 345-375.

Carey, T. (1989). Position paper: The basic HCI course for software engineers. *SIGCHI Bulletin*, *20* (3), 14-15.

Curtis, B. (Ed.) (1985). *Human factors in software development* (Second Edition). Washington, DC: IEEE Computer Society Press.

Curtis, B. (1987). Human-computer interaction: Special topic - The psychology of software development. *A graduate seminar in the department of management science and information*, Graduate School of Business, The University of Texas, Course Syllabus, Spring 1987.

Freeman, J. T. (1992). Conference report empirical studies of programmers: Fourth workshop. *SIGCHI Bulletin*, *24* (3), 18-23.

Hewett, T. T. (1987). An undergraduate course in software psychology. *SIGCHI Bulletin*, *18* (3), 43-49.

Hoc, J.-M, Green, T. R. G., Samurcay, R. & Gilmore, D. J. (Eds.) (1990). *Psychology of Programming*. London: Academic Press.

Lammers, S. (1986). *Programmers at work – Interviews*. Redmond, WA: Microsoft Press.

Roe, A. (1977). *The psychology of occupations*. New York, NY: Arno Press.

Sackman, H. (1970). *Man-computer problem solving: Experimental evaluation of time-sharing and batch processing*. Princeton, NJ: Auerbach Publishers.

Shneiderman, B. (1980). *Software psychology: Human factors in computer and information systems*. Cambridge, MA: Winthrop Publishers.

Shneiderman, B. (1986). No members, no officers, no dues: A ten year history of the software psychology society. *SIGCHI Bulletin*, *18* (2), 14-16.

Shneiderman, B. (1992). *Designing the user interface* (2nd edition). Reading, MA: Addison-Wesley.

Soloway, E. & Spohrer, J. C. (Eds.). (1989). *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Vaske, J. J. & Grantham, C. E. (1990). *Socializing the human-computer environment*. Norwood, NJ: Ablex Publishing Co.

Vat, K. H. (2000). Teaching software psychology: Expanding the perspective. *Technical Symposium on Computer Science Education: Proceedings of the thirty-first SIGCSE technical symposium on Computer science education,* Austin, Texas, United States, 392-396.

Weinberg, G. M. (1971). *The psychology of computer programming*. New York, NY: Van Nostrand Reinhold.

Weinberg, G. M. (1982). *Understanding the professional programmer.* Boston, MA: Little, Brown and Company.

Biography

John Lenarcic is a Physicist and Applied Mathematician by training, an Information Technology Academic by fortunate accident and an Armchair Philosopher by conscious choice. His research interests include social informatics and he is a frequent radio and television commentator on the societal and ethical aspects of IT. Lenarcic currently ruminates at length on IT matters at large as a Lecturer in the School of Business Information Technology at RMIT University in Melbourne, Australia.