STATUS OF COMPUTER AIDED SOFTWARE ENGINEERING TOOLS AND EDUSET EDUCATIONAL SOFTWARE ENGINEERING TOOL

Ben Livson
Ultraware Limited
P. O. Box 367
Kiryat Ono 55102
Israel

INTRODUCTION

Why are new computer science graduates almost useless as software engineers? Why must companies invest in massive retraining of CS graduates? Light is shed on these problems, and at a least a partial solution is attempted by the Educational Software Engineering Tool EduSET project.

Software engineering tools are supposed to aid, formalize or support the most difficult of all processes, namely the human reasoning process called software engineering. This process is by nature distributed, cooperative and heterogeneous. Interacting with project management, configuration management, software quality assurance, customer independent verification and validation is a most complicated, costly and painful process, even for experienced professionals. CS students in most cases do not have the faintest idea about these real life problems. EduSET simulates the real working environment, and lets students act in the roles of: developer; software quality assurance, configuration and project management; and customer end user. EduSET gives the framework for controlled access, with each user category having its privileges. EduSET enables the CS course software engineering lecturer to define a class project with students participating in the different roles. EduSET supports experimentation of life-cycle paradigms.

In this paper, the current state of software engineering is summarized. Minimal requirements for an 'Educational Software Engineering Tool' EduSET are presented. Advances and past disappointments with CASE computer aided software engineering tools are discussed and IEEE CS efforts summary is given. This paper provides a pragmatic outlook of what can be attained by computer aided software engineering tools.

CURRENT STATE OF SOFTWARE ENGINEERING

The view of Frederick P. Brooks, Jr. [1] that there is no magic 'Silver Bullet' to solve software engineering problems is widely accepted by both software engineering tool producers and users. Division of complexity into 'accidental' complexity related to the technicalities of the programming process, and to 'essential' problem complexity is used to bring the message that past breakthroughs have reduced accidental complexity but left the hard core problem of tackling the essential complexity. In particular, Brooks states that high-level programming languages, object-oriented programming, workstations, unified programming environments and tools for language support have reduced accidental complexity, and future gains are likely to be marginal in these fields. Program verification and automatic programming seem to be restricted to very narrow application areas. The relationship between artificial intelligence and software engineering is fuzzy, and attempts to use the now popular expert system technology to support software engineering need years to mature. Brooks lists ways to reduce essential complexity:

- Reusable software (Buy software; do not construct it)
- Requirements refinement and rapid prototyping
- Incremental development (Grow, don't build, software)
- Great designers (educational .. manpower issue beyond our scope)

COMPUTER AIDED SOFTWARE ENGINEERING 'CASE' - THE PROMISE ?

The bulk of software engineering is in the field of commercial data processing and management information systems. The hottest conference topic of 1987 and probably for the next few years is CASE; see [2] for an introduction. CASE systems support:

geometry. She started with the simplest programs dealing with the geometry of triangles and the associated theorems. The students could draw these diagrams and "see" the proof on rather rudimentary graphic systems (e.g., low resolution monochromic PC systems). Programs dealing with polygon geometry and the associated theorems were attempted but not completed during the one semester. She returned to teaching.

Case Study 2: Another student attempted to write programs dealing with molecular equations and valency bonds of interacting chemicals for her chemistry class. She attempted to explain the nature of the rudimentary chemical processes to her chemistry students. Her ambition (a commendable one) was to extend the programs to explain biochemical functions with a large database of the molecular representation for amino acids and their transformations in a biological system. This particular student was a highly skilled teacher herself, but decided to abandon her teaching to become a computer scientist. Her decision to pursue her new career is typical of about seventy percent of the teacher-graduate students. This leads to the our next discussion regarding our training programs for teachers.

The Scenario: The entry to the computer science programs opens such unexpected opportunities for the students that only a small percentage return to teaching. The methodologies are intricate and demand a dedicated scholarship to master them. Herein lies the paradox. The recently achieved mastery permits the present student to become a more highly paid employee in the computer industry or to become a researcher. Thus, the motivated teacher becomes a more highly paid computer programmer.

A Proposed Strategy: From the discussion with the students, the problem is not without an effective solution. The school administration should introduce man-machine communication and computer graphics as an early topic of study in the curriculum of the teachers. The teachers have to learn that interactive computer graphics is a potential and perhaps inevitable medium of instruction in the nineties. Sending bright uncommitted teachers to take a course in computer graphics amounts to issuing the teachers a free exit permit from their profession.

The use of computer graphics systems should be included during the formative years of the teacher education. The teacher then accepts this facility as human speech or a blackboard in the classroom. Thus the newer technology becomes firmly ingrained as a mode of teaching rather than as an alternative career path for the teachers. The proposed strategy suggests that the computer graphics instructors be implanted in teacher training colleges rather than sending teachers to computer science departments of local universities. In the former case, exposure to the subject will be in the context of teaching the younger students.

III. CONCLUSIONS

Interactive computer graphics as a mode of communication is extremely valuable to the teaching profession. Ignoring it can become futile, if not catastrophic. The loss to the teaching profession is comparable to the loss to the medical profession if doctors ignored computerized tomography techniques. Basic graphics interfaces for computer assisted instruction (CAI) via the visual channel in the early years of the teacher training program is suggested.

The amount of expertise necessary to use these techniques can be reduced enormously by developing teacher-friendly software to go with the powerful graphics chips embedded in personal computers. If the CAI packages are coupled with the graphics software, interactive learning can be individualized to the student needs, thus accelerating the "visualization" of concepts. In modern society where technical sophistication is the key to economic survival, students will be better adept at keeping up with the ever changing demands during their careers.

REFERENCES

1. M. S. Elphick, editor, "Microprocessor Basics", Hayden Book Company, Rochelle Park, N. J., 1977.

2. Intel Corporation, "80386 Hardware Reference Manual", Intel Literature Sales, Santa Clara, CA, 1986.

3. NEC, uP77230 Product Release for the Application of DSP for high quality, high speed graphics. Also see HP, Model 320S Graphics Workstation incorporating Graphics Processing Functions for Scan Conversion, FPP, and Pixel Caching.

4. von Nuemann, J., "Collected Works" edited by A. H. Traub, Volume 5, "Design of Computers, Theory of Automata and Numerical Analysis", Pergamon, New York, 1963.

- graphics programming
- prototyping, usually with the aid of an applications generator
- central data dictionary .. relational database environment
- code generator, usually part of 4G application generators
- software design methodologies such as Yourdon, Warnier-Orr or Chen entity relationship.

It is interesting to note that Brooks does not believe in graphics programming for visualizing software. CASE technology represents an attempt to adopt semi-formal software engineering methods for commercial data processing, a thing that has not succeeded in embedded computer systems engineering. However, it should be remembered that most commercial data processing is in stable and perfectly understood application domains, rendering them suitable for application generator specification. A significant effort for the past decade has been to embed software engineering in relational database management systems with central data dictionaries. These data bases cover project management information, configuration management, and the interlinked requirements and acceptance testing databases. Non-ambiguity, traceability, testability, and management visibility are breakthroughs that are attainable by incorporating DBMS technology into software engineering. Additional benefits are increased reusability of software, better cost and reliability estimation.

Crucial in CASE technology are user interface management systems [3]. The share of software dedicated to user interface management has increased from ten percent to more than seventy percent. The user interface management is the major gain for software engineering from advanced microcomputer and workstation technology.

CASE is an outgrowth of advanced DBMS and 4G application generator technology. Its solutions mainly address the Brooks accidental complexity problem. It is unlikely that CASE technology will have a major impact on embedded computer systems software engineering in the near future. Some two percent of commercial data processing facilities benefit from CASE solutions.

THE IEEE COMPUTER SOCIETY'S TASK FORCE ON TOOLS

Robert Poston [4] has proposed establishment of a task force on professional tools for the follcwing two initial activities:

1: On-line data-base for tool vendor entries organized around the IEEE Taxonomy of Software Engineering Standards [5]. The importance of this activity is paramount to enable tool users to have an easy access to public domain tools data that is both timely and well organized.

2: Tool interface standards. IEEE requirements, design and testing standards [5] shall for the first time address the interfacing of requirements definition tools to both design and testing tools. A standard data base schema is proposed for development tool interfaces.

The IEEE task force on tools could achieve the above goals within the next five years with profound implications.

DIGITAL EQUIPMENT VAXSET -- AN ANATOMY OF A SET OF TOOLS

DEC VAXSet [6] software productivity tools evolution is outlined to project future trends in software tools.

Phase-1: The first step of a computer vendor is to provide low level configuration management tools. AT&T Bell UNIX was the first operating system to provide a source code control system to support version control and a make utility to generate systems according to file update time dependency relations. Digital released its Code Management System and Module Management System to cover the basic area of low level configuration management.

Phase-2: The DEC Language Sensitive Editor was Digital's answer to an integrating editor, compiler and debugger. The great contribution of the AI community was to start integration of basic programming tools some fifteen years ago. These solutions are now available even for microcomputer users with Microsoft Corporation offering an integrated compiler, interpreter, debugger and Make as part of its new compiler releases. Additional tools such as the DEC Source Code Analyzer are basically extensions to compiler technology and are part of the Minimal Ada Programming Support Environment (MAPSE) toolset requirements [7]. The UNIX Lint C verifier is one of the earliest and most widely used static code analyzers.

Phase-3: DEC software testing tools include the DEC Test Manager, an autoregression test tool that is for testing what Make is for configuration manage-

ment. The Performance Coverage Manager is the DEC answer to performance tuning and dynamic analysis needs. It should be noted that both DEC SCA as a static analysis tool and PCA as a dynamic analysis tool were preceded by a number of public domain research systems with much more ambitious goals; see [8] for a discussion about the US TOOLPACK project as an environment for scientific programming and for references about the earliest static and dynamic analysis tools. The attractiveness of the PCA and SCA tools is that they are available for all DEC VAX/VMS programming languages and support all language extensions by this vendor.

Phase-4: The DEC software Project Management System is the follow-on of project management systems released for microcomputers and as such is a product that was conceived a number of years ago. A number of tool environments are centered around or include a project management system [9].

Phase-5? DEC CASE for VAXStations with structured analysis and design tools including editors for dataflow diagrams, structure charts, data structures and entity-relationships. An early solution in this field is IDE by Interactive Development Environments Inc.

Phase-6? Large scale integration of VAXSet with Vax Information Architecture VIA tools such as Rdb and CDD. Integration with application generators such as DEC Cobol generator.

Phase-7? DEC User Interface Management System.

Phases 5-7 are likely to commence within the next five years.

MINIMAL REQUIREMENTS FOR AN EDUCATIONAL SOFTWARE ENGINEERING TOOL

The concept of a software environment driver is implemented by the Ultraware Development Management System [10]. Minimal requirements for a software environment based EduSET are:

1: Provide a REUSABLE LIBRARY of driver functions, thus enabling customization and expansion into a full-scale environment.

2: Provide a CALLABLE INTERFACE. The interfacing options include:

- hidden driver menu interface option to user library functions
- access to global variables of the driver
- access to meta-data of the software engineering process
- standardize on the IEEE tool interfacing [4].

3: Implement and enforce a LIFE-CYCLE PARADIGM. Standardize on the IEEE software life-cycle paradigms [11].

4: Provide SOFTWARE ENGINEERING TEMPLATES. Implement the IEEE templates and provide a facility to implement military standard and corporate software engineering standard templates.

5: Provide a USER INTERFACE MANAGEMENT SYSTEM. This may be implemented as functions of the reusable driver library and as an application generator for user interface management software.

6: Provide AUTOMATIC LOW LEVEL CONFIGURATION MANAGEMENT FUNCTIONS such as version control and project history database.

7: Support PROJECT MANAGEMENT of MILESTONES and DELIVERABLE ITEMS.

8: Check the SANITY of the SOFTWARE ENGINEERING PROCESS. Do a cross check between project management data and what is implemented as a physical file system.

9: Reduce response time to less than the human threshold of 100 milliseconds. This is critical for the acceptance of automatic low level configuration control operations in real time.

10: Support corporate life-cycle paradigm definition and enforcement.

11: Support HUMAN COMMUNICATIONS and PROJECT INTEGRATION. In life, very few worthwhile things are done alone.

R E F E R E N C E S

[1] F.P. Brooks, "No Silver Bullet", IEEE Computer, April 1987.

[2] R. Hurst, "CASE System Near Fruition", Computerworld Focus, July 1987.

[3] S. Kolodziej, "User Interface Management System", Computerworld Focus, July 1987.

[4] R. Poston, "Charter for the IEEE Computer Society's Task Force on Professional Tools", IEEE CS Software Engineering Standards Subcommittee status report, August 14, 1987.
[5] IEEE Standard Taxonomy for Software Engineering Standards, IEEE Std 1002-1987.
[6] VAX/VMS Software, Language and Tools Handbook, Digital Equipment Corporation 1987.
[7] Requirements for the Programming Environment for the Common High Order Language, STONEMAN, Department of Defense, Washington, D.C., February 1980.
[8] W. R. Cowell and W. C. Miller, "The TOOLPACK Prospectus", Argonne National Laboratory, TM 341, September 1979.
[9] M. Dowson, "ISTAR - An Integrated Project Support Environment", ACM SIGPLAN Notices, vol. 22 no. 1, January 1987.
[10] B. Livson, "Development Management System", 1987 DECUS Symposium 2057, Rome, Italy, September 1987.
[11] IEEE P1074 Standard for Software Life Cycle Processes,to be balloted in 1989.

REQUEST EduSET REFERENCE READINGS

[1] Ben Livson, Future Software Development Management System Concepts, ACM SIGSOFT Software Engineering Notes, volume 12, number 3, July 1987.

[2] DMS Development Management System Driver Software, Software Engineering Environments, New.

[3] Software Development Paradigms and DMS, New.

Companies or individuals interested in making a submission to EduSET may contact Ben Livson, EduSET author, Ultraware Limited:

+972-3-341366, P.O. Box 367, Kiryat Ono 55102, Israel.
USA Telex: 4900004722 and INC Dialcom E-Mail 05:GLU750.

(Dr. Ben Livson is the co-founder and president in Ultraware Ltd, an Israel-based firm that provides training and consulting services and develops software tools with special emphasis on software quality assurance, testing and configuration management.)

READER CORRESPONDENCE

COMPUTER CRIME - MYTH AND REALITY

This is the name of a lecture, an article and maybe more than only an article, that I am trying to write these days, as a part and a basis for a new area of academic research and a new approach towards computer crime and criminology ("white collar crime" in computerized environment).

I am looking for written material and any information you can supply concerning these subjects. Any kind of information, books, newspaper articles, publications, reports, theses etc. - will be very much appreciated.

Some people think that the computer did not change basic concepts of criminology and did not affect basic criminal facts. These people ignore the change that took place after the "computer revolution" and the huge spread of PC's all over the world. It was the change of environment, the change of the criminal type (not the type of the crime itself), and in the boundary of the fraud as a new problem in the computer environment.

Before the computer revolution, the "white collar crime" was based on techniques of changing and fraud of documents. Today, the main crime is usually based on the deformation of computerized information. Any attach and/or destruction of any computer resource is equal to some type of old fashioned crime.

Is the computer crime a myth for users and "big money" for consultants? Are the ideas written before, the real and main point of these subjects? Is it possible to compare old fashioned crime to computer crime and find similarities in fraud type and in the criminal type?

Could you please send me any available information regarding these subjects and the subject of "myth and management"?

Thank you in advance.

Sincerely yours,

M. Peleg
P. O. Box 3352
Haifa 31032
ISRAEL

CHINA from page 1

a very important role as an economic leverage to keep the national macroeconomy under control, to ensure sufficient financial support to the State's "Four modernization" projects and to encourage the development of planned commodity economy in addition to offering ordinary banking services. Obviously, in order to achieve these purposes, ICBC must completely reform its organization structure, business operations and data processing methods by using computer facilities and electronic data processing technologies to replace the old manual labour and obsolete operating methods. The headquarters of ICBC has made an ambitious plan to computerize all its branch banks over the country and to build a nationwide computer network to link them together by the year 2000. Recently, as the first step of the project, ICBC has imported dozens of computer systems from IBM, HITACHI and other vendors, which have been installed and put to work in many important cities of the country.

Need for Trained Personnel

It is clear that to furnish banks with modern computer facilities is one thing and to actually computerize banks is another thing. One of the most urgent and key tasks of ICBC for its computerization is staff education in order to qualify them to operate, maintain and manage the installed computer systems efficiently. ICBC has more than 400,000 employees and, at the present time, most of them are completely unfamiliar with computer technology; therefore, the education of the staff of ICBC for its computerization is really a very tremendous and difficult job.

In order to realize the computerization plan of ICBC, we need professional people at different levels, such as: the manager of a computer center of a branch bank; software systems analyst; software systems manager or administrator; data processing expert; computer communications and networking engineer; hardware field-service engineer; power supply and air conditioning technician; programmer; and a great number of operators and tellers. Education for the higher level professional people usually requires a relatively long time period (e.g. 2 or 3 years) to learn all fundamental science and computer professional courses. Generally, it takes the form of full-time schooling. The medium level professional people, because of their large numbers, cannot expect to obtain many opportunities for education in full-time schools. However, they can be educated by spare-time or part-time courses from TV Open University, various night schools, correspondence schools and even by self-study. These are more efficient and economical ways to educate the staff members although they may need a longer time. Sometime, depending on the demands of the bank, we will also provide some short-term training classes to train people such as programmers and hardware maintenance technicians. The training of lower level people like terminal operator tellers usually is provided by the branch offices themselves.

Staff Education at ICBC

Our institute, the Hangzhou Institute of Financial Managers of ICBC, as an important staff education institute, is now taking on four different levels of education for the staff members of ICBC:

1) College-level full-time diploma education;
2) Continuing education;
3) Post training; and
4) Short-term professional training courses.

<u>College-level, Full-time, Diploma Education</u> (2 or 3 years) Specialties:

a) Applications software engineering

On successful completion of the three (or two) year education program, the student will be able to design, program and maintain the applications software used for banking services. A diploma will be offered to the qualified student by the Institute.

b) Computer hardware engineering

On successful completion of the three (or two) year education program, the student will be able to maintain the various computer hardware facilities, to evaluate the performance of computer systems and to recover the system when faults happen.

<u>Continuing Education</u>

With the rapid development of computer science and technology and the banking business as well, those who are involved in data processing and bank computerization must renew their knowledge their knowledge and skills continuously to keep up with the pace of bank modernization.

The working computer data processing engineers and other professional people can handle the daily transaction processing, but they are often too busy to study new knowledge and technologies,