Needs of the Software Industry in the Next Decade

Tuncer I. Ören
University of Ottawa
Department of Computer Science
Ottawa, Ont. K1N 6N5
oren@csi.uottawa.ca
(613) 564-5068

So far as users are concerned, the aim of computerization is not necessarily to develop software but to solve problems with the assistance of computers. Therefore, the software industry would serve the users better by providing Computer-Aided Problem Solving (CAPS) environments. Teaching software packages as well as advanced Computer-Aided Software Engineering (CASE) tools and environments can be useful as part of short-term vocational training. However, what is important is the realization of the necessary shifts of paradigms in software engineering in order to establish computer-aided problem solving as a reliable, easy, efficient, productive, and profitable process.

It is already a well established fact that most of the application code should not be written by humans. Code generators should replace the labor-intensive and error-prone manual code generation process (Bell Canada 1992, Lowry 1991). Such an attitude necessitates problem specification environments where problems can be specified, consistency checks can be done by knowledge-based tools, and code can be generated by an appropriate code generator. In this approach, a user can concentrate on the problem to be solved, rather than instructing a computer how to do what needs to be done. An added advantage is the direct re-usability and maintainability of the problem specifications. This approach may also assure built-in quality in computerization, eliminating expensive software testing activities which can not add quality but may only help remove defects implanted due to the deficiencies of manual code generation processes.

About a decade ago, a Japanese company (Olympus) was asserting that "unintelligent computerization is not enough." Currently not only cameras but some other machines benefit from such an attitude. For example, an intelligent copier which was announced recently in Japan has the ability of self-diagnosis and self-repair. These abilities are of course realized by software which provide advanced knowledge processing abilities to such machines. When we consider the needs of the software industry, we should consider software for stand alone computers as well as for intelligent machines which may involve autonomous, semi-autonomous, and other machines such as household machines, machine tools, robots, airplanes, missiles, space crafts, etc.

Similar to intelligent machines, one needs advanced software with goal-directed, adaptive, learning and self-tailoring capabilities. Some of the potential benefits of Knowledge-Based

Software Engineering (KBSE) or applications of Artificial Intelligence in Software Engineering (AISE) are already well proven. Not to consider AISE would simply result in delaying the imminent benefits of the synergy of the two fields (KBSE-92, Lowry 1991, Ören 1990). Some applications would result in the automation of the computerization process. Some other practices may be useful for existing software. For example, a recent program understanding schema developed under the supervision of the author can be used to display knowledge over 150 aspects of an object-oriented C++ software system. The schema can be beneficial for automatic software redocumentation as well as for software maintenance (Pankaj 1993).

Most of the software systems are perfect examples of complex systems as discussed in systems theories and in systems engineering. Since these scientific fields are best suited to study complex systems, and not only to cope but to monitor and manage the associated complexity, it is highly desirable to develop the synergy of software engineering and systems engineering to converge on software systems engineering. Already there is the beginning of such a trend. For example, in a leaflet dated November 1992, Elsevier Science Publishers state:

> "The field of computer programming has developed from a programming-oriented field towards a more system-oriented field. As a result of this development the journal Science of Computer Programming has adapted its aims and scope."

Software systems engineering can benefit from Computer-Aided Systems Technology (CAST) as well as Computer-Based Systems Engineering (CBSE) which are related recent trends to provide computer-based tools to systems engineering in general.

In systems engineering, user/system interfaces are studied in depth under ergonomics. Similarly, principles, methodologies, techniques, and tools for software ergonomics should be developed, possibly benefitting from artificial intelligence techniques.

Trillium (Bell Canada 1992) which is inspired of the Carnegie Mellon University's Software Engineering Institute Capability Maturity Model incorporates the intent of several international standards such as ISO, IEEE, IEC, and Bellcore standards. It has the potential to become the international standard for a software development capability assessment model. This excellent work can benefit from additional research in two directions. One is the refinement of the criteria in the fourth and especially fifth level of the Computer-Aided Problem Solving capability. The other is the development of necessary computer-aided tools to transform this set of evaluation criteria into a set of action-oriented recommendations associating specific people in an organization with tasks to be performed to reach and maintain a given level of proficiency. The last issue of maintaining a certain level of software development ability is equally important, since similar to flying an airplane, once a certain satisfactory state is reached, one cannot afford to stop the activities.

References

Bell Canada (1992). Trillium - Telecom Software Product Development Capability Assessment Model (Draft 2.2), July 1992.

KBSE-92 (1992). Knowledge-Based Software Engineering. Proceedings of the 7th Conference, Sept. 20-23, 1992, McLean, VA. IEEE CS Press, Los Alomitos, CA.

Lowry, M.L. (1991). Software Engineering in the Twenty-First Century. In: Automating Software Design, M.L. Lowry and R.D. McCartney (eds.). AAAI Press / The MIT Press, Menlo Park / Cambridge / London, pp. 627-654.

Ören, T.I. (1990). A Paradigm for Artificial Intelligence in Software Engineering. In: Advances in Artificial Intelligence in Software Engineering - Vol. 1, T.I. Ören (ed.), JAI Press, Greenwich, Connecticut, pp. 1-55.

Pankaj, B. (1993). A Program Understanding Schema for Object-Oriented C++. Master's Thesis. University of Ottawa, Systems Science Programme, Ottawa, Ontario.