Looking for the AI in Software Engineering An Applications Perspective

Mark S. Fox

Center for Integrated Manufacturing Decision Systems
Carnegie Mellon University
Pittsburgh, PA 15213

msf@cs.cmu.edu

Introduction

I have been asked to address the reasons why Artificial Intelligence has not had a significant affect on real world Software Engineering, from the perspective of the development of smart application components. In the following I will explore twhat it means to be a "smart application component", then explore why they have not had the impact Robert Balzer thinks they should deserve.

Smart Application Components

What is a smart application component? The noun "component" implies that it is a software module or procedure that will be attached to or embedded in another, larger piece of software. Consequently, the component plays a supporting role and must conform to the environment in which it is embedded. The adjective "smart" implies that the component is able to perform some decision task, or participate in one such that the final decision is "better". Lastly, the adjective "application" implies that the component's functionality is specific to a domain and not general across all domains. The question is "do we have any?"

Of course, knowledge engineering software systems abound, such as ART™, KEE™, and Knowledge Craft™. They provide a knowledge representation, usually frame-like, a rule language, usually OPS5-like, and a variety of software engineering support functions such as editors, debuggers, and browsers. These systems have built upon the concepts found in interactive LISP environments and have clearly influenced the direction of the software engineering field. But they are at too low level of functionality to satisfy the application criterion mentioned above. Methods for truth maintenance, assumption-based truth maintenance andconstraint satisfaction are now common place and easily accessed, but these too are at too low a level of functionality.

A number of commerical packages focusing on solving diagnosis or troubleshooting problems have been developed. The earliest being derivatives of EMYCIN from Stanford, such as, Cimflex Teknowledge's S.1™, and M.1™ and Texas Instrument's Personal Consultant™. TestBench™ from Carnegie Group is a more recent example of a troubleshooting shell that uses failure mode analysis to identify sources of malfunctions. In the latter case, the representation and problem solving capabilities are restriected to a narrower domain.

A generalization of the above has evolved into the concept of a *generic task,* proposed by (Chandrasekaran, 1988). The belief is that software can be designed by configuring a set of tasks, such as:

* Hierarchical classification
* Hypothesis matching
* Synthesis by plan selection and refinement
* Abductive hypothesis assembly

While an interesting conjecture, it is too early to tell what such a library would be composed of, and there exists too little experience in configuring them.

Lastly, more specific application components have been created, some of which are available commercially. These include, knowledge based simulation packages such as Simulation Craft from Carnegie Group and Simkit from Intellicorp, factory scheduling packages, process planners, mechanical design systems, etc.

It is clear that Artificial Intelligence has begun to produce smart application packages, but only recently.

Embedability

Given that there exist components that could potentially be embedded in another system, what has curtailed their use? The answer is LISP. Historically, LISP could not be run as an embedded module within a larger application. Either its need to be the "master" process, and/or its size has been the problem. While many believe that if this problem is solved, these components will find greater use, the percpetion is false. The real problem is that industry does not want to train their employees to program in LISP, no matter what the AI community perceives its benefits to be. Training programmers in a new language is expensive, and managers loathe spending large amounts on learning what is essentially a fringe language with questionable, in their eyes, benefits. The momemtum that exists in industry today is to standardize on a single operating system, such as UNIX, and a single programming language, such as C/C++.

Change is in the air. Neuron Data's NEXPERT^{TM} is C-based and embedable. The recently announced IMKA consortium, composed of Digital Equipment, Ford Motor, Texas Instruments, USWest, and Carnegie Group has as its goal to create a C-based knowledge representation system and library of application representations that are efficient, embedable and portable.

Reusable Representations

One would think that with the central role that knowledge representation plays in Artificial Intelligence, that there would be a greater availability of representation libraries. It is true that the last ten years has witnessed a flurry of work in temporal and spatial representations, so that we now have several to choose from. And in some areas, such as factory scheduling and project management an attempt has been made towards crating libraries (Sathi et al., 1985). Only Lenat's CYC project at MCC has pushed the representational frontier, and more recently, DARPA, NSF and AFOSR convened a workshop to promote the reusability of representations. Never the less, AI application builders tend to build their representations from scratch.

Conclusion

It is clear that there are not many AI-based smart application components available. It follows then that they should have little impact on software engineering. On the other hand, what modules are available tend to be written LISP, crippling their adoption by industry. The only positive statement one could make, is that software engineering has yet to achieve the "holy grail" of resuability either.

References

Chandrasekaran, B., (1988), "Generic tasks as building blocks for knowledge-based systems: the diagnosis and routine design examples", *The Knowledge Engineering Review*, Vol. 3, No. 3, pp. 177-182.

Sathi, A., Fox, M.S., and Greenberg, M., (1985), "Representation of Activity Knowledge for Project Management", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 5, pp. 531-552.