Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

Artificial intelligence education for software engineers

B. Lees
Department of Computing and Information Systems, University of Paisley, Paisley, PA1 2BE, UK

Abstract

This paper presents a case for educating software engineering students in artificial intelligence methods. The relationships and interactions between software engineering and artificial intelligence and the characteristic nature of artificial intelligence methodology are discussed. Aims for an AI course for software engineering students are outlined. The structure of such a course and a strategy for teaching the subject are discussed, that the author has used successfully with undergraduate software engineers.

1. Introduction

Artificial intelligence (AI) and software engineering have developed, for the most part, along different lines, with different aims. But AI has much to offer software engineering, and vice versa. From its early beginnings AI has paved the way in the provision of substantial support for program development via interactive debugging aids and powerful program development environments. Furthermore, various software engineering programming practices, for example, list processing, hierarchical design and program modularisation, have their origins in the course of early explorations into what would be needed in order to make a computer behave in an intelligent way[1].

It is very desirable that students who will become the software engineers of the future be provided with the opportunity of learning about those aspects of AI which may be of value in the development of large software systems, or which are likely to become so in the future. The purpose of such studies is not to attempt to make them into AI specialists, but, rather, to enable them to realise when the incorporation of AI methods may be beneficial in the construction of software systems.

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

2. Inter-Relationships Between AI and SE

Three different ways in which artificial intelligence and software engineering may interact may be considered:

(i) *The use of AI supported tools for software systems development* - i.e. the use of smart software development environments. Such tools may be directed towards broad issues, for example requirements analysis and design[2], or to support specific aspects of software development, for example quality assessment[3].

(ii) *The software engineering of AI systems* - i.e. the adoption of proven techniques and methodologies in the construction of large, complex AI systems, for example intelligent knowledge based systems.

(iii) *AI supported software systems* - i.e. the inclusion of artificial intelligence components, for example automated reasoning, natural language processing, or the embodiment of AI techniques in software systems, so as to extend the power and capabilities of the resulting system.

The first two of the above are complementary, in that in (i) it is AI that is used to assist the process of software engineering, whereas in (ii) it is software engineering methodology that is used, as a basis for developing AI systems. In (iii), the integration of AI and software engineering is aimed at producing more powerful systems than would be possible in the absence of the AI components.

3. What is Artificial Intelligence?

3.1 Scientific and Engineering Views of AI

Two main activities of AI may be identified[4]:

- designing new, useful intelligent or semi-intelligent machines,
- the empirical study of existing intelligent systems (mainly humans) through the creation of computational models.

The first of these may be considered to be the engineering approach to AI, whereas the second may be regarded as the scientific approach. This scientific approach is largely the domain of cognitive psychologists. Both approaches are important for the advancement of AI; they are mutually supportive. However, it is the former of these, the engineering approach, which is of greater interest and importance in the study of AI from the perspective of a software engineer.

In an alternative analysis, there are considered to be three types of AI[5]:

- *Basic AI* - in which computational techniques are explored that have the potential for simulating human intelligence.
- *Modelling human or animal intelligence, using AI techniques* - this is the scientific orientation of AI as identified above

- *Applied AI* - the use of existing AI techniques for industrial or commercial applications

3.2 Analogy with Mathematics and Engineering

Bundy compares computing science and AI with engineering and science[6]. In his view, computing science is similar to mathematics, basic AI is similar to applied mathematics, and applied AI is similar to engineering. Both mathematics and computing science provide a range of techniques for developing theory and for modelling. Applied mathematics is used for building models in the natural sciences; whilst basic AI is used to study, for example, methods of inference and knowledge representation, that have been found to be of use in modelling aspects of intelligence. In applied AI these techniques are then used to build industrial or commercial software products. In pure engineering (or engineering science) new techniques are developed, e.g. for electrical circuit synthesis, which can then be used in engineering, for the solution of practical problems; similarly, AI techniques, developed in basic AI, e.g. constraint propagation, can be used in applied AI.

3.3 AI and Computing Science

In most academic institutions, artificial intelligence is usually considered as an area of computing science. But AI is also heavily dependent on concepts and theories drawn from other disciplines, notably psychology, philosophy and linguistics. It should therefore not be thought that AI techniques are a subset of computing science techniques, for, in fact, it may be argued that AI provides a wider range of methods than does computing science. In computing science and software engineering generally, the basic approach is largely algorithmic. The particular strength of AI lies in the provision of alternative problem solving methods which may be effective in the solution of problems for which algorithmic methods fail, or for which no algorithms are known. Furthermore, various programming paradigms are also employed in AI: in addition to algorithmic methods, functional programming, logic programming and object-oriented programming are all used. Thus, in certain situations AI is able to provide additional problem solving power over and above that provided by computing science. Since software engineering is also largely based on the algorithmic computing science, it is to be expected that AI may be able to enhance software engineering also.

3.4 AI Methodology

The main methodology in AI is that of exploratory programming[7]; to model a new situation, a simple prototype is developed, then tested with further examples. The ensuing results will then suggest possible inadequacies in the model and point to ways in which the program may be extended. Along the way, some analysis of the processes in the program will be carried out. Starting from a domain specific method the aim is to generalise the program. If this is possible the aim should be to put the program on a more formal basis by means

of a specification, and by verifying the technique, exploring relationships in other domains and testing on standard examples. Bundy notes that these latter, more formal stages are not, always widely practiced, but, nevertheless, are very important and warrant greater attention[8]. The adoption of these stages brings AI methodology closer to the formal approach of software engineering

The solution of an AI problem cannot, in general, be specified in the same tight sense as that of a software engineering problem. AI programs are usually non-deterministic: there may be several possible solutions, any one of which may be acceptable. But, unlike software engineering, the exact sequence of steps required to obtain a solution will, in general, not be known.

4. Comparison of AI & Software Engineering

4.1 Methodologies

From a software engineering standpoint, software development may be considered as a process which starts by creating a formal specification, that is then successively refined, to produce an algorithm which is then (ideally) formally verified or, at least, exhaustively tested. In contrast to this, AI problems are usually not well-defined, thus precluding the formulation of a formal specification. Consequently, the AI approach does not involve the design of an algorithm to implement a specification. This leads Partridge to define AI *as "the design of tractable approximations to intractable specifications"*[9].

The view of Partridge and Wilkes is that the natural paradigm for the development of AI systems is an iterative sequence of: *run - understand - debug - edit*, rather than the *specify - prove - implement - verify* approach of software engineering[10].

In the application of AI to the design and construction of expert systems, rapid prototyping has proved to be an effective developmental approach; a small-scale system is quickly developed, with the aim of obtaining a deeper understanding of the problem, as a preparation for the specification and construction of a full-scale version. Much of the activity centres on the handling of knowledge. Buchanan *et. al.*[11] have identified that the process of developing a knowledge-based system comprises the following stages:

identification:	determining problem characteristics
conceptualisation:	finding concepts to represent knowledge
formalisation:	designing structures to organise knowledge
implementation	formulating rules that embody knowledge
testing:	validating rules that embody knowledge

The overall process is an iterative one and, in general, will involve several successive re-formulations, re-designs and refinements.

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

4.2 The Problem of Verification and Validation

The task of verifying an AI program takes on a different form from that of software engineering. Because of the non-deterministic, non-algorithmic nature of the subject, a rigorous formal verification, as in software engineering, is usually not possible. But one can take another approach. For example, in a knowledge-based system the knowledge base can be validated to check that there are no inconsistencies, for example conflicting production rules. AI techniques have been developed for handling such situations, e.g. truth maintenance. Also it is possible to verify the basic inference mechanism of the system. Although the verification of knowledge-based systems is less well-developed than other aspects of knowledge engineering, methods have been developed which are based on the detection of anomalies or incompatibilities, or on statistical analysis of aspects of production rules[12]. What it is not normally possible to do is to verify all possible interactions among the inference mechanism, the knowledge base and all possible arrangements of the working memory throughout the running of the program. The situation assumes even higher levels of complexity when the knowledge base is dynamic, or when concurrency is employed.

Because of these difficulties, the certification of software systems which include knowledge based elements, particularly for safety critical systems, is problematic. As a result, there are problems associated with the inclusion of non-algorithmic software in closed-loop control systems[13].

5 Artificial Intelligence for Software Engineers

5.1 Why Software Engineers Should Learn AI

Software engineering and AI are mutually beneficial. Software engineers should have a knowledge of AI because (i) AI can assist software engineering, and (ii) software engineering can assist AI.

AI can assist software engineering: by enhancing the power of software systems, providing them with added functionality, or by providing solutions when other methods fail, for example by making intelligent use of large amounts of knowledge (which may either be structured or unstructured). The most important benefit that a study of AI can offer a software engineer is a widening of his/her repertoire of problem solving skills. Because some computer users may find it easier to specify a problem in natural language, the provision of special-purpose natural language user interfaces is desirable. AI is required here to advance the natural language processing capabilities of software systems. AI also involves methods for the representation of conceptual knowledge from which inferences may be drawn through

computation. The issues involved in knowledge representation are common to many domains.

Software engineering can assist AI: If software engineers have a knowledge of AI they will be able to play a valuable role in the introduction of improved methods for the development of AI software. Support for the need of more software engineering in AI is provided by Irgon *et. al.* who, in a retrospective analysis of the development of five separate expert systems, concluded that the most difficult and time-consuming issues are often non-AI issues, for example, the designing of user interfaces, and the development and integration with conventional software[14].

5.2 Course Aims

The purpose of an AI course for software engineering students is not to make them into AI specialists, but to provide them with a knowledge of the capabilities and limitations of current knowledge-based methods and technology to enable them to assess when the application of such an approach may be feasible and beneficial. Suggested aims of a course in artificial intelligence for software engineers are:

- to widen the student's range of computer-based approaches to problem solving, beyond purely algorithmic methods,
- to study the concepts and techniques of knowledge-based systems and their construction,
- to demonstrate when and how a knowledge based approach can be used to enhance software engineering systems,
- to provide practical experience in the construction of knowledge based systems,
- to encourage the student to employ a software engineering approach in the development of programs which incorporate knowledge-based techniques,
- to enable the student to identify where the application of knowledge based techniques may be appropriate for the solution of a problem.

5.3 A Strategy for Teaching AI

The central theme underlying a first course in AI for software engineers is the use of a variety of problem solving approaches. Most AI systems are based on symbolic programming methods, for which special-purpose languages, e.g. Lisp and Prolog, have been devised. But in recent years an alternative approach, connectionist AI, i.e. the use of artificial neural networks, has emerged. The course of study should include both symbolic and connectionist AI.

Course Content

The predominant method of AI problem solving involves heuristic search: inference methods involving various types of search should be studied. Additional problem solving methods for study are: constraint satisfaction, case-

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

based reasoning, model based reasoning, means-ends analysis, genetic algorithms and neural networks. The students also need to study various knowledge representation formalisms, e.g. production rules, frames and scripts, as well as the structure of knowledge-based systems. To provide students with a basis for the construction of intelligent user interfaces, syntactic and semantic modelling aspects of natural language processing should be included. The student should learn to program in an AI language, be introduced to additional AI programming paradigms and, if possible, gain some experience in interfacing AI components with other software. If time allows additional application areas, e.g. machine vision, speech recognition, AI in engineering, can be covered.

Teaching Approach

A practically-based course is necessary, for which an expert system shell is unsuitable as it would not allow the necessary low-level access to enable students to experiment with underlying inference mechanisms and knowledge structures. It is important that students gain some experience in AI programming. But AI systems are often large and complex and, because of time constraints, students can not be expected to program, from scratch, many systems based on the range of problem solving approaches that are being studied.

An approach which the author has found to be effective, is to provide a set of software modelling blocks for use by students[15]. The language used is Prolog, which has the advantage of allowing the construction of very concise programs to illustrate, at an appropriate level of abstraction, the concepts being studied. Students learn to program in Prolog and then use the software modelling blocks to construct solutions to problems provided in structured worksheets; they are also required to modify certain blocks to match the needs of other problems. The modelling blocks supplied support various search mechanisms, knowledge representation schemes and natural language processing.

6. Discussion

In the UK there has been rather slow acceptance of the use of AI in commercial software. This contrasts with the situation in North America and in France, where there has been a greater willingness on the part of software companies to invest in AI applications. To a certain extent the reasons for this are perhaps historical; the Lighthill Report in the nineteen seventies had the effect of putting the brakes on AI work in Britain and imposed a constraint to which AI researchers in the US and France were not subjected. However, there now exists a range of problem solving approaches that have emerged from AI research laboratories. The integration of AI and software technologies promises exciting possibilities for the future through: increasing use of hybrid AI systems, parallel processing support for AI, intelligent agents in distributed systems, and intelligent extraction of knowledge from data. The present

Transactions on Information and Communications Technologies vol 10, © 1995 WIT Press, www.witpress.com, ISSN 1743-3517

generation of software engineering students needs to be adequately prepared for the challenges posed by such developments.

References

1. **Simon H A**. Whether software engineering needs to be artificially intelligent, *IEEE Transactions on Software Engineering*, vol SE-12 (7), July 1986, pp 726-732.
2. **Puncello, P** *et al.* ASPIS: a knowledge-based CASE environment, IEEE Software, March 1988, pp 58-65.
3. **Hamza, M**, *et. al.* Quality function deployment for software development: an A.I. approach, *Procs. TQM94*, AMTA, Alexandria, Egypt, April 1994.
4. **Sloman, A** *Intelligent Systems: a Brief Overview,* Cognitive Science Research Paper, CSRP 027, The University of Sussex, 1983.
5. **Bundy, A**. What kind of field is AI? In D. Partridge and Y. Wilkes (eds) *The Foundations of AI: A Sourcebook,* Cambridge University Press, 1990.
6. *ibid.*
7. *ibid.*
8. *ibid.*
9. **Partridge, D** Do we really want machines to be intelligent? *Procs Software Engineering Practices for Intelligent Knowledge Based Systems*, Digest No: 1989/43, IEE, London, March 1989.
10. **Partridge, D and Wilkes, Y**. Does AI have a methodology different from software engineering? In D. Partridge and Y. Wilkes (eds) *The Foundations of AI: A Sourcebook,* Cambridge University Press, 1990
11. **Buchanan, B.G**. *et. al.* Constructing expert systems. In: F. Hayes-Roth, *et. al.* (eds) *Building Expert Systems* , Addison-Wesley, 1983.
12. **O'Keefe, R M and O'Leary, D E**. Expert system verification and validation: a survey and tutorial, *Artificial Intelligence Review*, vol. 7, 1993, pp 3-42.
13. **Cullyer W.J**. Certification of safety-critical systems, *Procs Software Engineering Practices for Intelligent Knowledge Based Systems*, Digest No: 1989/43, IEE, London, March 1989.
14. **Irgon, A**. *et. al.* Expert system development: a retrospective view of five systems, *IEEE Expert,* June 1990, pp 25-40.
15. **Lees, B.** Prolog support tools for learning AI concepts, Procs. Workshop on Logic programming and Education, ICLP'94, Santa Margherita, Italy, June 1994, pp 45-50.