Reflections on being an AI System Architect

Victor Lesser

June 2008

Traditionally, AI researchers have been attracted by the idea of creating machines that are intelligent, or understanding of the nature of human intelligence. My path to becoming an AI researcher was nontraditional, and in a way, this path has shaped my view on what the important issues are in AI. My computer science Ph.D. dissertation was not in AI or a related field, but in parallel computer architectures and operating systems [1] — yet, in a surprising way its origins are indirectly linked to AI. The idea for my dissertation grew from a term project that I did in a computer architecture course that Professor John McCarthy taught when I began my graduate studies at Stanford in the fall of 1966.

AI at the Forefront of Software Complexity

It was probably inevitable that I would eventually do research in AI since my abiding attraction to computer science has been the opportunity for constructing complex hardware and software artifacts and understanding how they operate. Since its inception, AI has pushed the boundaries of software complexity in order to construct programs that exhibit intelligence. Essential to the AI enterprise, and probably the aspect that has caused its software to be complex from the very beginning, is the need to deal directly with uncertainty in its many guises—uncertainty in: the information that is sensed from the environment; the situation specificity of knowledge (there is rarely common-sense knowledge that can be applied uniformly over a broad spectrum of real-world situations); the lack of complete theories that fully explain naturally occurring phenomena; and the bounded rationality of computation (NP hardness) that makes it impossible to fully assess all options. Dealing with the ubiquitous uncertainty in both data and control has led to ideas such as delayed resolution of uncertainty through non-deterministic and assumption-based computation, asynchronous and opportunistic application of knowledge and its associated constraints, exploiting approximate knowledge and heuristics to focus problem-solving activities, combining diverse sources of knowledge to resolve uncertainty, and self-aware computation and meta-level reasoning to provide more context for decision making under uncertainty.

Having uncertainty as a first-class concept in computation, and the related need for large amounts of knowledge (and its modular representation and flexible application), have caused AI to constantly go beyond the contemporaneous boundaries of software complexity. This will likely continue in the future as the field builds increasingly advanced intelligent systems that operate in more open, distributed and real-world environments. The architectural design and control of such intelligent systems in terms of defining the components that make up the system and specifying the mechanisms and protocols that define their interaction has been an important aspect of my research.

The Beginnings of My Career in AI

I feel fortunate to have launched my AI career as the system architect on the Hearsay-II speech-understanding project [7] at its inception in late 1972 at CMU, under the leadership of Professor Raj Reddy[1], along with other team members including Lee Erman, Rick Fennell and Rick Hayes-Roth. The ideas that developed around this project and the lessons learned in building this blackboard problem-solving system have been extremely influential in the research directions I have explored in my career. These laid the groundwork for my understanding of the complex interaction of knowledge and search control necessary to bring to bear in an effective way a diverse and large set of constraints to resolve a variety of uncertainties in knowledge, data and control. In the case of a blackboard problem-solving architecture, this interaction took the form of an asynchronous, opportunistic, incremental and multi-level search process.

Self-Aware Control and Self-Describing Components

In controlling the blackboard search process, I came to understand the need for control mechanisms that could make use of an abstract and global view of the search progress so as to determine where search should be directed next, and that the control in itself was a problem-solving activity [3, 15]. I also realized that the specific type of knowledge-source modularity in the blackboard architecture in which knowledge sources made very few assumptions about how their knowledge contributions would be used by other components and in what context they would be used was a powerful idea. This context modularity made possible not only ease of system evolution but also adaptive problem solving where the control system can react effectively to the evolving problem-solving system state by being able to dynamically sequence components. For instance, this permitted Hearsay-II to find alternative paths to the correct speech interpretation if a correct partial solution path failed because of a knowledge error.

A key aspect of Hearsay-II's approach to modularity was the self-describing nature of the knowledge sources' activity patterns (both static and dynamic)[2]. This information could then be used for effective control without the need for understanding how and what knowledge would interact prior to the construction of the control component [6].

---

[1]My transition to AI from systems was not as abrupt as it may appear. My initial AI work at CMU had a systems oriented flavor; I was responsible for implementing a parallel processing version of Hearsay-I on a multi-processor C.MMP that had just been developed at CMU [2]. In fact, one of the initial focuses on the design of Hearsay-II was its ability to be run on a multi-processor [5], which is probably why I was chosen as a member of the Hearsay-II design team. Additionally, I flirted with AI earlier when I worked for Professor Reddy in the summer of 1967 looking at how techniques used in information retrieval could be applied to pattern classification problems in AI. Prior to that, I had worked for Professor Gerald Salton on information retrieval during my senior undergraduate year at Cornell.

[2] I had used a similar idea of self-describing computation in a more primitive form in my Ph.D. thesis, where I included a declarative representation of the application process structure to help in the hardware scheduling of processes on processors.

Dynamically Varying Assumptions

For efficiency reasons, the number of hypotheses on the blackboard needed to be limited in the Hearsay-II implementation; however, it was clear that these limits (thresholds on hypothesis generation) were context-dependent and had to be modified selectively if problem solving was not progressing as expected. Modifying these thresholds caused less likely hypotheses to be generated that in turn permitted the exploration of other problem-solving paths that were initially considered unlikely [3].

This need for dynamic thresholds led me to understand that the large number of assumptions (which are implemented as system parameters) that are made in controlling a problem-solving system must be made explicit in order for the system to reason about these assumptions. Exploiting assumptions is a valid way of restricting the complexity and overhead of control decision-making in a problem-solving system, but as problem solving gets more complex and the environment in which the system operates is more open, these assumptions need to be revisited in light of both the emerging problem-solving situation and the current environment. Associated with revision of assumptions is the need to be able to detect that problem solving is not progressing satisfactorily, diagnose why it is not progressing, revise assumptions that are no longer valid, and then re-plan how to approach problem solving based on these new assumptions [11, 21].

Diverse and Approximate Knowledge

I also learned from my experiences with Hearsay-II the value of diverse and approximate knowledge. The diversity of knowledge increases the flexibility of the system to find alternative way of solving a problem. When there are errors or incompleteness in one source of knowledge it is possible that another source of knowledge will be able to generate the appropriate solution. Multiple perspectives also provide additional error correcting capabilities in terms of the system being able to compare alternative hypotheses generated from different knowledge perspectives.

The use of approximate knowledge, which in general requires much less computation to exploit than more exacting knowledge, was also an important part of Hearsay-II's implementation. This approximate knowledge was used for self-aware control [3] and hypotheses generation [4] to direct the search in appropriate directions. Furthermore, approximate knowledge can often be used to generate complete solutions (though not necessarily optimal nor with perfect precision) that have significant utility with correspondingly much lower computational costs [29].

Multi-Agent Systems Pushing the Boundaries of AI Software Complexity

Multi-Agent Systems (first called Distributed AI) were initially developed out of applying AI problem solving in a distributed context (such as a robotic team or distributed sensor network) where there was a spatially distributed set of computers (now called agents) — possibly with local sensing and effecting capabilities, operating asynchronously and

concurrently — that could talk with one another over a low-bandwidth communication network. This bandwidth limitation was most interesting to me because it not only introduced the need for distributed control, but also increased the amount of uncertainty present in the system because it was no longer possible to accurately construct a complete view of the problem-solving activities (and their associated results) of all the agents in the network. The development of distributed techniques for managing and resolving problem-solving uncertainty, including both data and control, has been one of my central concerns since the completion of the Hearsay-II speech understanding project in 1976[4] [8, 9, 12, 13, 25, 28].

I have purposely used the term "managing uncertainty" to recognize the fact that in AI (and especially multi-agent) systems there will always be problems that cannot be solved optimally (nor in every situation will the correct answer be chosen) due to limitations based not only on the incompleteness and inaccuracy of the system's knowledge, but also on computation and communication limitations (bounded rationality) [9, 16]. Thus, Simon's idea of satisficing problem solving was, and to my mind, always will be, central to the AI enterprise. These observations have led me to study: 1) how the interaction of diverse sources of knowledge in a distributed search can be used to manage and resolve uncertainty, and 2) how self-aware local control and agent coordination can be used to control this distributed search. In developing this multi-agent control, I have viewed control as a first-class problem-solving activity, made it as generic as possible through self-describing agents, and enabled it to be adaptable in response to emerging problem-solving conditions and available resources.

The Need for Experimental AI

In performing research in multi-agent systems (MAS), I have always emphasized building and empirically evaluating complex applications. Again, this orientation has risen from my experiences with the Hearsay-II system. I have found that dealing with details and constraints of real applications has continually pushed the development of new ideas. In such applications, you cannot easily sweep problems under the rug. Although there can be significant start-up costs in knowledge engineering a real-world application, in my experience those costs have been worthwhile. Although sometimes I have needed to reduce the complexity of the problem domain, this step has come after first tackling the harder problem and realizing how and why current approaches fail. Recently, I worked with Professor Shlomo Zilberstein on developing more formally based approaches to agent coordination using decentralized Markov decision processes [24]. In this work, I recognized the value of starting with a problem of simpler complexity and then addressing more complex problems as the formal framework and the appropriate solution techniques are better understood. I suspect there is no single best approach for all situations, and the right approach depends in part on the personality of the researcher and the specifics of the problem.

[4]I also worked on a more advanced blackboard problem-solving architecture for understanding overlapping household sounds, called IPUS [17]. This architecture used many of the insights described above to allow for dynamic and adaptive signal reprocessing based on a detection and diagnosis model for meta-level control.

Experimentation is More than Gathering Statistics

With the ever-increasing availability of computing cycles, extensive empirical evaluation of systems is feasible. However, it takes more than gathering statistics about the performance of the system. Understanding in detail why a system works or not in a specific problem-solving instance has continually provided insights that have pointed me to new research directions. Probably the most compelling example was my realizing, by looking at the trace of a Hearsay-II run, that even though there was a serious knowledge error, there was enough redundancy in the knowledge base and flexibility in the search process for the correct solution to be derived. This experience, and a follow-on experience with a parallel version of Hearsay-II (that worked almost correctly without all the elaborate synchronization mechanisms that had been constructed to avoid inconsistency when there was concurrent updating of the blackboard), led to my work on multi-agent systems [5]. I had the insight that agents did not require complete, up-to-date and consistent information for their local problem solving to be worthwhile. Even if they were not always producing correct partial solutions, they could still converge on the correct solution most of the time by combining an appropriately structured cooperative dialogue that implemented a flexible distributed search process together with some level of knowledge and data redundancy distributed across the agents [8, 9].

Thus, as an AI system architect I have been concerned about how to add both internal and external mechanisms to the application so that it is possible to quickly understand the detailed functioning of the system, not only for debugging but also for understanding how it works [6, 10, 22]. I believe that as we continue to build increasingly more complex applications, these "system understanding" tools will be crucial to their successful implementation.

Learning as an Integral Part of the System's Architecture

In a similar vein, it is important to consider on-going learning as a first-class activity of any system that will operate in an open environment [19, 20]. This is especially true in complex applications where there are many parameters (with their associated assumptions about the environment and problem-solving behavior) that have to be set appropriately if the system is to operate effectively in a specific environmental setting. However, the computational costs of this on-going learning may not permit it to be always turned on. Thus, the decision of when to perform learning should be made by the control component [21].

The Challenges of Scaling up Multi-Agent Systems

One approach that I have been pursuing for the next generation of complex AI applications is using organizational control to scale up multi-agent systems to large networks of hundreds to thousands of intelligent agents [23, 28]. Organizational control is a multi-level control approach in which organizational goals, roles, and responsibilities are dynamically developed, distributed, and maintained to serve as guidelines for making detailed operational control decisions by the individual agents. Scaling up through organizational control is one way of approaching the development of highly intelligent and persistent systems that will

bring to bear a wide range and large body of knowledge to their reasoning and that will operate in open and evolving environments.

Formalizing our Understanding of AI Architectures

AI architectures used in complex applications have generally had an informal character and their performance has been evaluated from an empirical perspective. Based on the paradigm shift in AI towards more formal and decision-theoretic problem-solving frameworks and detailed analysis of their performance characteristics, it is clear to me that this shift will also affect how complex AI applications are designed and understood. It does not necessarily mean that the informal character of AI architectures for complex applications will disappear; however, their performance has to be understood from a more formal basis. This trend is already occurring. There are already numerous case studies where the performance of a sophisticated multi-agent system involving multi-step knowledge interaction sequences has been effectively analyzed from a quantitative perspective using statistical and queuing theory models based on a deep understanding of the key parameters and interactions that affect performance [14, 27].

It was very rewarding to see the application of formal analysis techniques to a blackboard problem-solving system that many now see as an ad-hoc problem-solving architecture. This analysis predicted the performance of the system, while also providing insight into which situations a particular control regime would be effective in, and why. This work was done by one of my students, Robert Whitehair [18], for the specific task domain of sensor interpretation. Thus, I believe that the performance of less formally based AI architectures can be understood more formally, thereby gaining wider acceptance by AI researchers as valid approaches for building complex AI applications.

Likewise, the recent work on formalizing agent coordination through decentralized Markov decision processes [24] has been very exciting. It has shown, in general, how hard multi-agent coordination is, and how easy it is to find simple applications that are very hard. This implies that some amounts of heuristic and approximate reasoning need to be applied to use this formal framework in practical applications. These observations coincide with those of Professor Milind Tambe [26] on the importance of hybrid architectures as an approach to building complex intelligent systems. These hybrid systems are built from some components that use formal decision theoretic methodologies, while other components use methodologies more symbolic and possibly heuristic in character. The blackboard problem-solving framework is an early example of a hybrid architecture because each knowledge source could be built with its own specialized technology. Indeed, in the Hearsay-II speech understanding system, most knowledge sources were heuristic, but a key knowledge source for doing word recognition used a hidden Markov model for its internal reasoning.

Closing Thoughts

I see AI architects as indispensable to the effective construction of future complex and open applications because of both the satisficing nature of AI computation and the range of different types of uncertainty that will be present. However, I do think that the arsenal of both

formal and informal tools, frameworks and methodologies that they will use in developing and understanding the intelligent systems they create will be much more sophisticated and extensive than the ones that I have used.

On a personal note, being an AI system architect has led to a highly rewarding and exciting career, challenging me intellectually at every step. I have no doubt that the next generation of AI architects will have similar experiences. I have also been very fortunate to have worked with wonderful teams of colleagues and graduate students. Without their creativity, hard work and collegiality, my career would have taken a very different path. I also need to give special thanks to my mentor, Professor Raj Reddy, who started me down the path of AI and encouraged me to pursue the area of multi-agent systems.

Background Papers

1. Lesser, V.R. (1971). "An Introduction to the Direct Emulation of Control Structures by a Parallel Micro-Computer." *IEEE Transactions on Computers - Special Issue on Micro-programming,* C-20, 7, pp. 751–763.
2. Lesser, V.R. (1975). "Parallel Processing in Speech Understanding Systems: A Survey of Design Problems." *Invited Papers of the IEEE Symposium on Speech Recognition,* D.R. Reddy (ed.), pp. 481–499.
3. Hayes-Roth, F.; Lesser, V.R. (1977). "Focus of Attention in the HEARSAY-II System." *Proceedings of the 5th International Joint Conference on Artificial Intelligence,* Boston, pp. 27–35.
4. Lesser, V.R.; Hayes-Roth, F.; Birnbaum, M.; Cronk, R. (1977). "Selection of Word Islands in the HEARSAY-II Speech Understanding System." *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing,* Vol. 2, pp. 791–794.
5. Fennell, R.D.; Lesser, V.R. (1977). "Parallelism in AI Problem Solving: A Case Study of HEARSAY-II." *IEEE Transactions on Computers - Special Issue on Multiprocessors,* C–26, 2, pp. 98–111.
6. Erman, L.D.; Lesser, V.R. (1978). "System Engineering Techniques for Artificial Intelligence Systems." *Computer Vision Systems,* A. Hanson and E. Riseman (eds.), Academic Press, pp. 37–46.
7. Erman, L.D.; Hayes-Roth, F.; Lesser, V.R.; Reddy, D.R. (1980). "The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty." *Computing Surveys* 12, (2), pp. 213–253.
8. Lesser, V.R.; Erman, L.D. (1980). "Distributed Interpretation: A Model and an Experiment." *IEEE Transactions on Computers - Special Issue on Distributed Processing,* Vol. C–29 (12): 1144–1163.
9. Lesser, V.R.; Corkill, D.D. (1981). "Functionally Accurate Cooperative Distributed Systems." *IEEE Transactions on Systems, Man, and Cybernetics - Special Issue on Distributed Problem-Solving,* Vol. SMC–11, No.1, pp. 81–96.
10. Bates, P.C.; Wileden, J.C.; Lesser, V.R. (1983). "A Debugging Tool for Distributed Systems." *Proceedings of the Second Annual Phoenix Conference on Computers and*

Communications, pp. 311–315.

11. Hudlickà, E.; Lesser, V. (1987). "Modeling and Diagnosing Problem-Solving System Behavior." *IEEE Transactions on Systems, Man, and Cybernetics,* 17(3): 407–419.
12. Durfee, E.H.; Lesser, V.R. (1991). "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation." *IEEE Transactions on Systems, Man, and Cybernetics,* 21(5): 1167–1183.
13. Lesser, V.R. (1991). "A Retrospective View of FA/C Distributed Problem Solving." *IEEE Transactions on Systems, Man, and Cybernetics,* 21(6): 1347–1362.
14. Decker, K.; Lesser, V. (1993). "An Approach to Analyzing the Need for Meta-Level Communication." *Proceedings of the 13th International Joint Conference on Artificial Intelligence,* Vol. 1, pp. 360–366.
15. Carver, N.; Lesser, V. (1994). "The Evolution of Blackboard Control Architectures." *Expert Systems with Applications—Special Issue on the Blackboard Paradigm and Its Applications*. Liebowitz (ed.). New York: Pergamon Press, 7(1): 1–30.
16. Sandholm, T.; Lesser, V. (1995). "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework." *Proceedings of First International Conference on Multi-Agent Systems,* San Francisco: AAAI Press, pp. 328–335.
17. Lesser, V.; Nawab, H.; Klassner, F. (1995). "IPUS: An Architecture for the Integrated Processing and Understanding of Signals." *Artificial Intelligence,* 77, pp. 129–171.
18. Whitehair, R. (1996). "A Framework for the Analysis of Sophisticated Control." Ph.D. Thesis, Department of Computer Science, University of Massachusetts Amherst.
19. Sugawara, T.; Lesser, V.R. (1998). "Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments." *Machine Learning,* Kluwer, 33 (2-3): 129–153.
20. Nagendra Prasad, M.V.; Lesser, V. (1999). "Learning Situation Specific Coordination in Cooperative Multi-Agent Systems." *Autonomous Agents and Multi-Agent Systems,* Kluwer Academic Publishers, Vol. 2, pp. 173–207.
21. Horling, B.; Benyo, B.; Lesser, V. (2001). "Using Self-Diagnosis to Adapt Organizational Structures." *Proceedings of the Fifth International Conference on Autonomous Agents,* ACM Press, pp. 529–536.
22. Lesser, V.R.; Corkill, D.D. (1983). "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks." *AI Magazine,* Vol. 4 (3): 15-33.
23. Horling, Bryan; Lesser, Victor (2004). "A Survey of Multi-Agent Organizational Paradigms." *The Knowledge Engineering Review,* Cambridge University Press, Vol. 19:4, pp. 281–316.
24. Becker, R.; Zilberstein, S.; Lesser, V.; Goldman, C. (2004). "Solving Transition Independent Decentralized Markov Decision Processes." *Journal of Artificial Intelligence Research,* Vol. 22, pp. 423–455.
25. Lesser, V.; Decker, K.; Wagner, T.; Carver, N.; Garvey, A.; Horling, B.; Neiman, D.; Podorozhny, R.; NagendraPrasad, M.; Raja, A.; Vincent, R.; Xuan, P.; Zhang, X.Q.

(2004). “Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework.” *Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers, 9 (1-2): 87–143.

26. Nair, Ranjit; Tambe, Milind (2005). “Hybrid BDI-POMDP Framework for Multiagent Teaming.” *Journal of Artificial Intelligence Research,* 23: 367–420.
27. Horling, Bryan; Lesser, Victor (2008). “Using Quantitative Models to Search for Appropriate Organizational Designs.” *Autonomous Agents and Multi-Agent Systems*. Springer-Netherlands, 16(2): 95–149.
28. Corkill, D.D.; Lesser, V.R. (1983). “The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network.” *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* pp. 748–756.
29. Lesser, V.; Pavlin, J.; Durfee, E. (1988). “Approximate Processing in Real-Time Problem Solving.” *AI Magazine,* Volume 9, Number 1, pp. 49–61.