Teaching Intelligent Agents: Software Design Methodology

LTC Michael Bowman, Dr. Antonio M. Lopez Jr., MAJ James Donlon
U.S. Army War College

Dr. Gheorghe Tecuci
George Mason University

Current software design and development methodologies have evolved slowly since the early days of computing and today almost universally include user-developer interaction for requirement determination, testing, and acceptance activities. A new paradigm for software development is emerging from U.S. government sponsored research in artificial intelligence for rapid knowledge-based systems development. This new paradigm, being pursued by the George Mason University Learning Agents Laboratory, calls for end users to interact directly with an intelligent software agent to develop their own problem solving intelligent agents. This approach is called Disciple, and it has been experimentally shown to rapidly produce high performance software agents for solving complex military problems.

Software design methodologies are intended to help develop computer programs more systematically. They are sets of procedures that people follow from the beginning to the completion of the software development process. Since the 1970s, the numbers of software design methodologies have increased significantly. Khoo [1] gives an overview of some of the major ones, and Sorenson [2] narrows the focus to a comparison of those methodologies used by Department of Defense software developers. In both presentations, the authors recognize that the selection of an appropriate methodology depends on a number of factors, including the type of problem being addressed and the qualifications and training of the people using the methodology.

It is universally recognized that software design is a creative process that cannot be reduced to a routine procedure; however, it is not devoid of structure. For the typical data processing or information derivation problem, the framework for software design and development is given in Figure 1.

The user specifies the requirements (usually poorly) and the software development team, after gathering data and clarifying some ambiguities, selects and uses a design methodology to develop the desired software product. The software development team may have many members, including systems analysts, senior and junior programmers, quality assurance personnel, and technical writers. The process is error prone.

As Humphrey [3] puts it: "The process is so complex because many different people are involved and they are all learning. At the outset, no one really understands either the requirements, the design, or the implementation. This is particularly true when the implementation finally progresses to the point where the users can first try the product (p. 313)."

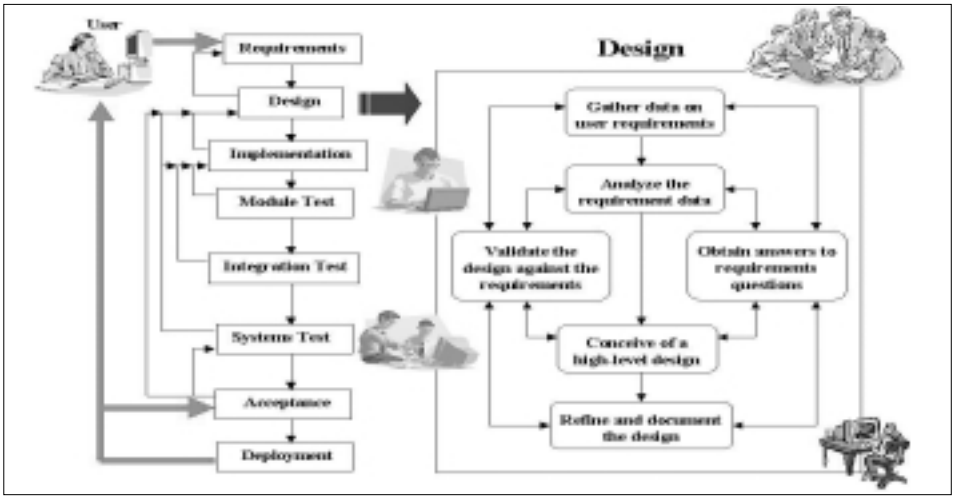
There is a class of difficult problems and a group of highly qualified and trained people who routinely solve such problems, but these people are not software designers or developers. We categorize the problems as knowledge-based problems and the people as subject-matter experts (SMEs). The military has SMEs in many critical areas, and they possess priceless knowledge in how to solve problems of great importance. The skills, insights, and creativity of such individuals make them true craftsmen.

For several decades artificial intelligence (AI) researchers have addressed the problem of developing knowledge-based software agents that incorporate the expertise of the SMEs and exhibit a similar problem-solving behavior. Generally, the agent includes two main components: a knowledge base and an inference engine. The knowledge base contains the data structures representing the entities from the expert's application domain such as objects, relations between objects, classes of objects, laws, actions, processes, and procedures. The inference engine consists of the programs that manipulate the data structures in the knowledge base in order to solve problems in a way that is similar to how the expert solves them.

Typically, developing an intelligent software agent that assisted or replicated the SME required the SME to work closely with a knowledge engineer who would actually build the agent [4]. In this case, the framework for the software development is analogous to that presented in Figure 1 with the software design effort of the knowledge engineer being based more on gathering and representing knowledge than on data and information. Here we view information as interpreted data, but knowledge is information in action or information transformed into capability for effective action [5].

The process of acquiring knowledge from an SME and representing it in the knowledge base of the agent has been found to be difficult and labor intensive, and is what has come to be known as the *knowledge acquisition bottleneck*. The process is slow and difficult because knowledge engineers and domain experts do not initially speak the same language. During the process of indirect knowledge

Figure 1: *A Typical Framework for Software Development*



transfer from the SME through the knowledge engineer into the agent's knowledge base, the SME and knowledge engineer must achieve a common understanding of the domain and how problems are solved in the domain. They must also produce a mutually understood representation of the domain and problem solving. There is in a sense, a cross leveling of language and expertise.

A Different Paradigm

A contradiction exists in the classical knowledge-based agent development process where the presence of the knowledge engineer is both part of the solution and part of the problem. An additional difficulty with this paradigm is that the development of each new knowledge base or agent starts from scratch, with no knowledge reuse in spite of the fact that knowledge acquisition is such a difficult process. The Learning Agents Laboratory (LALAB) at George Mason University (GMU) is developing a new approach to agent development that addresses these problems and significantly alters the framework in Figure 2.

The goal is to enable an SME that does not have prior knowledge engineering experience to develop a knowledge-based agent by himself or herself, with limited or no support from a knowledge engineer. This approach, called Disciple [6], is based on developing a very capable learning agent that has two main characteristics: 1) it uses a type of knowledge representation and organization that facilitates knowledge reuse and learning, and 2) can be directly taught by an SME how to solve domain-specific problems in a manner that resembles the way the SME would teach a human apprentice – by giving the agent examples and explanations as well as by supervising and correcting its behavior.

Over the years, the LALAB has developed a series of increasingly more capable learning agents from the Disciple family. The general architecture of a Disciple agent is shown in the upper right side of Figure 2. The problem-solving engine is based on the general problem reduction paradigm of problem solving and is therefore applicable to a wide range of domains. In this paradigm, a problem to be solved is successively reduced to simpler problems until the problems are simple enough to be solved immediately. Their solutions are then successively combined to produce the solution to the initial problem.

The learning and knowledge acquisition engine integrates several learning strategies synergistically, such as learning from examples, learning from explanations, and learning by analogy, in order to acquire the knowledge from the SME. The knowledge base is structured into two distinct components: an object ontology and a set of reduction and composition rules. The object ontology is a hierarchical representation of the objects and types of objects from a particular domain, such as military or medicine. That is, it represents the different kinds of objects, the properties of each object, and the relationships existing between objects. The object ontology provides a representation vocabulary that is used in the description of the reduction and composition rules. Each reduction rule is an *if-then* structure that expresses the conditions under which a problem P_1 can be reduced to the simpler problems P_{11}, … , P_{1n}. Similarly, a composition rule is an *if-then* structure that expresses the conditions under which the solutions S_{11}, … , S_{1n} of the problems P_{11}, … , P_{1n} can be combined into a solution S_1 of P_1.

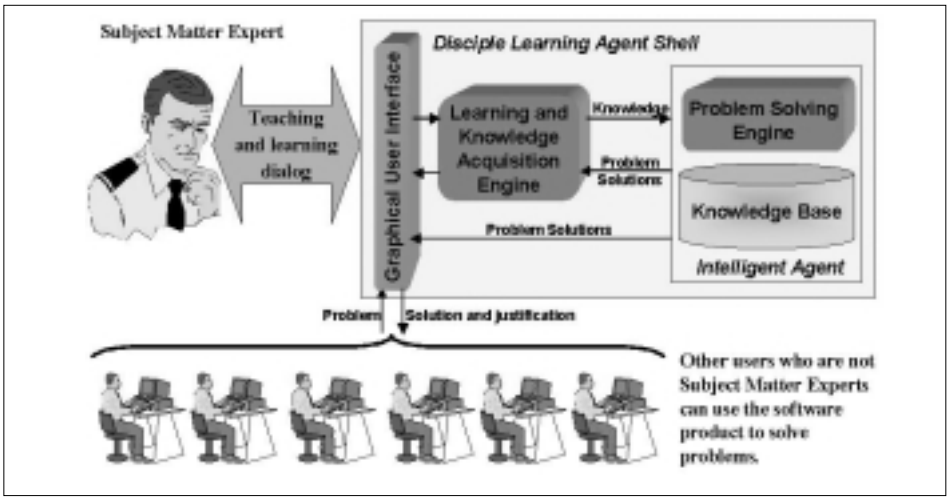
Dividing the knowledge base into an object ontology and a set of rules is very important, because it clearly separates the most general part of it (the object ontology), from its most specific part (the rules). Indeed, an object ontology is characteristic to an entire domain. In the military domain, for instance, the object ontology will include descriptions of military units and of military equipment. These descriptions are most likely needed in almost any specific military application. Because building the object ontology is a very complex task, it makes sense to reuse these descriptions when developing a knowledge base for another military application, rather than starting from scratch.

In the case of Disciple, the ontology reuse is further facilitated by the fact that the objects and the features are represented as frames, based on the knowledge model of the Open Knowledge Base Connectivity (OKBC) protocol. OKBC has been developed as a standard for accessing knowledge bases stored in different frame representation systems [7]. Therefore, importing an ontology from an OKBC compliant knowledge server, such as Ontolingua [8] or Loom [9], does not raise translation problems.

The rules from the knowledge base are much more application-specific than the object ontology. Consider, for instance, two agents in the military domain, one that critiques courses of action with respect to the principles of war, and another that plans the repair of damaged bridges or roads. While both agents need to reason with military units and military equipment, their reasoning rules are very different, being specific not only to their particular application (critiquing versus planning), but also to the SMEs whose expertise they encode. Therefore, the rules are generally not reusable from one application to another, and have to be defined for each new application.

Several decades of knowledge engineering attests that the traditional process

Figure 2: *A New Framework for Software Development*



by which a knowledge engineer interacts with a subject matter expert to manually encode his or her knowledge into rules is long, difficult and error-prone. The Disciple approach to rule development does not involve a knowledge engineer. Initially the SME teaches the Disciple agent how to solve problems by showing it each problem reduction step necessary to solve a specific problem, and helping the agent to understand it. From each such problem reduction step, the Disciple agent learns a general problem reduction rule that will allow it to perform similar problem reductions in the future.

As Disciple learns from the SME, their interaction evolves from a teacher-student interaction to an interaction where the SME and the agent collaborate in solving new problems. During this joint problem solving process, Disciple continues to learn new rules from the contributions of the expert, and to refine previously learned rules based on its own problem solving attempts.

Under this agent-building paradigm, knowledge engineers support the SME's creation of a specialized Disciple agent in these ways:

- By customizing the graphical user interface.
- By helping the SMEs to learn how to teach the Disciple agent.
- By facilitating the re-use of ontological knowledge found in established knowledge repositories such as the CYC knowledge base [10].

The ultimate software development framework for the SME-driven Disciple learning agent paradigm is given in Figure 2. This paradigm eliminates much of the error generated by the many different people involved in the typical framework for software development (Figure 1). Who knows the specific problem domain better than the SME – the requirements, the data, and the problem solving techniques?

Part of Disciple's output when it has solved a problem is an explanation of how it derived that solution. Thus other people who are not as familiar with the specific problem domain as the SME can use the trained version of Disciple to solve problems, understand the problem solving reasoning used, and learn themselves.

Because the problem solving approach of a Disciple agent is based on the general problem reduction paradigm, this agent-building methodology is applicable to a wide rage of domains. For instance, we have applied it to develop agents for the following:

- Planning the repair of damaged bridges and roads.
- Critiquing military courses of action.
- Identifying and testing strategic center of gravity candidates in military conflicts.
- Designing configurations of computer systems.
- Generating test questions for assessing a student's higher order thinking skills in history and in statistics, etc.

We consider that the main factors that contribute to the success of the Disciple approach are: 1) the synergistic collaboration between the SME and the Disciple agent in developing the knowledge base, 2) the multi-strategy learning methods of Disciple that are based on a learnable knowledge representation, and 3) extensive knowledge reuse. Nevertheless, this technology is still in its research phase, and significant work remains to be done before it will be available for mainstream DoD use.

Is software developed using an intelligent learning agent such as Disciple a new software design methodology? We claim that it is.

Recorded Successes

The Defense Advanced Research Project Agency (DARPA) has sponsored a sequence of two programs for the development of software systems that will solve knowledge-based problems. The first is the High Performance Knowledge Bases program (HPKB), which ran from fiscal year 1997 to 1999. The second, Rapid Knowledge Formation program (RKF) is currently funded for fiscal years 2000 to 2003. The goal of the HPKB program was to produce the technology needed to enable system developers to rapidly construct large knowledge bases that provide comprehensive coverage of military specific domains of interest, are reusable by multiple applications that use diverse problem solving strategies, and are maintainable in rapidly changing environments. The HPKB program used challenge problems to evaluate the competing software technologies.

In the first set of challenge problems, one of the problems dubbed *the workaround* problem consisted of assessing how rapidly and in what way a military unit might reconstitute or bypass damage to an infrastructure such as a bridge. The Disciple learning agent shell was customized to solve this problem. The SMEs supported by knowledge engineers identified many of the concepts that needed to be represented in Disciple's ontology (e.g. military units, engineering equipment, types of damage, and geographical features of interest). Some of these concepts were imported from existing ontology repositories. The SMEs and the knowledge engineer using Disciple's specialized browsing and editing tools defined others. Knowledge engineers also explained to the SMEs problem reduction modeling. This is an iterative process of stating a problem to be solved, asking a relevant question about solving the problem, and answering the question either conclusively, in which case that portion of the problem is solved, or by creating sub-problems that need further consideration.

We compare the problem-question-answer process to the Army's task-condition-standard testing methodology. Army SMEs had little difficulty in assimilating the idea. Further detail regarding the training inputs and reasoning outputs of Disciple for the workaround problem can be found in [11]. Cohen et al. [12] presents measured results for the first year of HPKB, including the workaround problem where the trained version of Disciple (Disciple-WA) was judged to perform at an expert level. It was also noted that Disciple-WA's knowledge actually doubled during the 17 days of experimental period, thus demonstrating the agent's ability to learn rapidly.

The second part of the HPKB program was based on even more complex challenge problems. The course of action (COA) critiquing problem called for rapid development of knowledge bases containing comprehensive battlefield knowledge. These include terrain characteristics, force structures, troop movements, and tactical strategies to assess COA viability, correctness, and strengths and weaknesses with respect to the principles of war and the tenants of Army operations. To address the challenge problem, the Disciple learning

agent shell was customized for the domain by knowledge engineers who imported an initial ontology built by Teknowledge and Cycorp, and by SMEs who further developed the ontology using Disciple's built in tools. The SMEs then taught Disciple how to evaluate a COA, and thus Disciple-COA was developed.

Tecuci et al. [13] presents the results of Disciple-COA evaluation vis-à-vis its competitors. In total scores on the metrics of recall and precision, Disciple-COA outperformed the competition. Furthermore, during the eight days of experimental period, Disciple-COA's knowledge increased by 46 percent, from the equivalent of 6,229 simple axioms to 9,092 simple axioms. This represents a higher daily rate of knowledge acquisition than in the first experiment.

To further test the ability of Disciple-COA to amass knowledge, it was used in a knowledge acquisition experiment at the U. S. Army Battle Command Battle Laboratory (BCBL) in Fort Leavenworth, Kansas. Four SMEs with no prior knowledge of engineering experience received approximately 16 hours of training on Artificial Intelligence and the use of Disciple-COA. Then using the knowledge base containing the previously developed ontology (but with no rules and no significant assistance from knowledge engineers), each SME taught Disciple to critique COAs. This was done with respect to a modeling of the principles of offensive and security that was discussed with them before the experiment.

In about three hours Disciple-COA learned 26 rules. Upon his evaluation, LTC John N. Duquette, chief of the Experimentation Division of BCBL, wrote [14], "The potential use of this tool by domain experts is only limited by their imagination – not their AI programming skills."

The Next Step

DARPA's follow-on RKF program emphasizes the development of knowledge bases by the domain expert. Its central objective is to enable distributed teams of SMEs to enter and modify knowledge directly and easily without the need for prior software development or knowledge engineering experience. The Knowledge Engineering Group (KEG) in the Center for Strategic Leadership at the United States Army War College (USAWC) recommended that the GMU LALAB be given the vexing Center of Gravity Determination problem as the RKF challenge problem for the next step in Disciple's evolution. The KEGs recommendation went to Murray Burke, the RKF program manager at DARPA, and was approved.

Joint Publication 3.0 Doctrine for Joint Operations [15] states, "Identification of enemy centers of gravity require detailed knowledge and understanding of how opponents organize, fight, make decisions, and their physical and psychological strengths and weaknesses. (pp. III-21)." In 1995 KEG researchers and SMEs in various departments at USAWC, along with interested USAWC students and International Fellows taking the elective course "Case Studies in Center of Gravity (COG) Determination," worked to capture the *current* thinking on determining strategic and operational centers of gravity. That thinking was published in a monograph with an accompanying process flow diagram [16]. Attempts by KEG researchers to build an intelligent system that could assist in center of gravity determination and analysis were constrained by the technology and theory of the day.

In the fall of 2000, KEG researchers began ontology development for Disciple-COG [17] and GMU LALAB researchers worked on new tools for Disciple that would aid SMEs in scenario building and domain modeling. In the winter 2001 term, USAWC students in the COG course used these tools to begin the process of building their own agent for the scenario that they have been given to analyze. In the spring 2001 term, another USAWC COG class will use the tools to analyze additional scenarios. Through a process of iterative refinement and with more examples being given to Disciple-COG with each new USAWC COG class, it is expected that the intelligent learning agent will become increasingly adept at determining and analyzing centers of gravity for opposing forces.

Conclusion

Software design and development methodologies have evolved slowly since the early days of computing. Today's methodologies call for extensive user-developer interaction for requirement determination, testing, and acceptance activities. New research and experimental results presented in this paper indicate that this may be about to change.

Supported by DARPA, the U.S. Air Force Office of Scientific Research, and the U.S. Army, the George Mason University Learning Agents Laboratory has taken a novel approach to the creation and use of intelligent agents to solve complex military problems. The goal of this research is to develop methods and tools that allow users with minimal computer skills to easily build, teach, and maintain high performance intelligent software agents. In this approach, the user teaches the agent to solve various problems in a way that resembles how they would teach an apprentice or student – providing examples and explanations, supervising and correcting behavior.

Created this way, the intelligent agent represents a new paradigm for software development. In this paradigm, the inefficient, often ineffective indirect transfer of knowledge and expertise from the end-user through the development team, to the resulting software product, is replaced by direct user development of the system, assisted by intelligent agent technology. This paradigm has already been experimentally shown to be effective in tackling selected military problem-solving at the tactical and operational level, and is now being evaluated at the strategic level.◆

References

1. Khoo, Benjamin, An Integrated Software Design Framework for the Design of Information Systems, Masters Thesis, Andrews University, Michigan, 1996. userpages.umbc.edu/~khoo/survey2.html
2. Sorenson, Reed, A Comparison of Software Development Methodologies, *CrossTalk*, January 1995, pp. 12-18.
3. Humphrey, Wyatt, *A Discipline for Software Engineering,* Addison Wesley, Reading, MA, 1995.
4. Russell, Stuart and Norvig, Peter, *Artificial Intelligence: A Modern Approach*, Prentice Hall Upper Saddle River, NJ, 1995.
5. Smith, Reid and Farquhar, Adam, The Road Ahead for Knowledge Management: An AI Perspective, *AI Magazine*, December 2000, pp. 17-40.

6. Tecuci, Gheorghe, *Building Intelligent Agents*, Academic Press, San Diego, CA, 1998.
7. Chaudri, Vinay; Farquhar, Adam; Fikes Richark; Park, Peter; and Rice, James, *OKBC: A Programmatic Foundation for Knowledge Base Interoperability*, Proceedings of the Fifteenth National Conference on Artificial Intelligence, 600-607, AAAI Press, Menlo Park, CA, 1998.
8. Farquhar, Adam; Fikes, Richard; and Rice, James, *The Ontolingua Server: A Tool for Collaborative Ontology Construction*, Proceedings of the Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Alberta, Canada, 1996.
9. MacGregor, Robert, Retrospective on LOOM, 1999, www.isi.edu/isd/LOOM/papers/macgregor/LoomRetrospective.html
10. Lenat, Douglas, CYC: A Large-scale Investment in Knowledge Infrastructure, *Communications of the ACM*, November 1995, pp. 33-38.
11. Tecuci, Gheorghe; Boicu, Mihai; Wright, Kathryn; Won Lee, Seok; Marcu, Dorin; and Bowman, Michael, A Tutoring Based Approach to the Development of Intelligent Agents, *Intelligent Systems and Interfaces*, Kluwer Academic Publishers, Boston, Mass., 2000.
12. Cohen, Paul; Schrag, Robert; Jones, Eric; Pease, Adam; Lin, Albert; Starr, Barbara; Gunning, David; and Burke, Murray, The DARPA High-Performance Knowledge Bases Project, *AI Magazine*, December 1998, pp. 25-49.
13. Tecuci, Gheorghe; Boicu, Mihai; Bowman, Michael; Marcu, Dorin; preface by Burke, Murray, An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a Course of Action Critiquer, *AI Magazine*, June 2001.
14. Bowman, Michael; Tecuci, Gheorghe; and Boicu, Mihai, Intelligent Agents, Tools for the Command Post and Commander, to appear in *Acquisition, Logistics, and Technology*, 2001.
15. Joint Publication 3.0, Doctrine for Joint Operations, Joint Chiefs of Staff, Washington, DC, 1995.
16. Giles, Phillip and Galvin, Thomas, Center of Gravity: Determination, Analysis, and Application, Center for Strategic Leadership, United States Army War College, Carlisle Barracks, Pa., 1996.
17. Bowman, Michael; Lopez, Antonio; and Tecuci, Gheorghe, Ontology Development for Military Applications, Proceedings of the Thirty-Ninth Annual ACM Southeast Conference, March 2001.

Acknowledgements

Research at the GMU Learning Agents Laboratory is sponsored by the DARPA, Air Force Research Laboratory, Air Force Material Command, USAF, under agreement number F30602-00-2-0546, by the Air Force Office of Scientific Research (AFOSR) under grant no. F49620-00-1-0072, and by the U.S. Army. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFOSR, the Air Force Research Laboratory, the U.S. Army or the U.S. government. Mihai Boicu, Dorin Marcu, Bogdan Stanescu, Catalin Balan, Elena Popovici, Cristina Cascaval, and other members of the LALAB contributed to successive versions of Disciple. The CrossTalk Editorial Board provided insightful comments and suggestions that helped us improve this paper.

About the Authors

LTC Michael Bowman is a student at the U.S. Army War College and a Ph.D. candidate at George Mason University. He was the Army product manager for Communications and Intelligence Support Systems, and has had a variety of acquisition, automation, and tactical assignments at the Defense Intelligence Agency, the U.S. Military Academy, and in several Army field artillery battalions. He received a bachelor's degree from Ouachita Baptist University and a master's degree from the Naval Postgraduate School.

Knowledge Engineering Group
Center for Strategic Leadership
United States Army War College
Carlisle Barracks, PA 17013
Voice: 717-245-3252
Fax: 717-245-4600
E-mail: michael.bowman@carlisle.army.mil

Antonio M. Lopez Jr., Ph.D., holds the chair of Artificial Intelligence in the Knowledge Engineering Group of the Center for Strategic Leadership at the U. S. Army War College. He also holds the Conrad N. Hilton Endowed Chair in Computer Science at Xavier University of Louisiana. He is a colonel (retired) in the U. S. Army Reserve having held assigned positions with the 377th Theater Support Command as assistant chief of staff G6, G3, and chief of staff.

Knowledge Engineering Group
Center for Strategic Leadership
United States Army War College
Carlisle Barracks, PA 17013
Voice: 717-245-3251
Fax: 717-245-4600
E-mail: lopezt@csl.carlisle.army.mil

MAJ James Donlon is the director of the Knowledge Engineering Group at the U. S. Army War College. He conducts applications engineering, education, and applied artificial intelligence research as an Army systems engineer. His Army background is as an infantry officer. He received a bachelor's degree from the University of Delaware and a master's degree from Northwestern University.

Knowledge Engineering Group
Center for Strategic Leadership
United States Army War College
Carlisle Barracks, PA 17013
Voice: 717-245-3265
Fax: 717-245-4600
E-mail: donlonj@csl.carlisle.army.mil

Gheorghe Tecuci, Ph.D., is professor of computer science, director of the Learning Agents Laboratory at George Mason University, and member of the Romanian Academy. He received his doctorate and master's degrees in computer science from the Polytechnic University of Bucharest, and a doctorate in computer science from the University of Paris-South. Tecuci has published more than 100 scientific papers and five books, most of them in the artificial intelligence areas of intelligent agents, machine learning, and knowledge acquisition.

Learning Agents Laboratory
Department of Computer Science
George Mason University
4400 University Drive
Fairfax, VA 22030
Voice: 703-993-1722
Fax: 703-993-1710
E-mail: tecuci@gmu.edu

"Computation offers a new means of describing and investigating scientific and mathematical systems. Simulation by computer may be the only way to predict how certain complicated systems evolve."

Stephen Wolfram