

# Rapid Development of Large Knowledge Bases\*

Marcel Barbulescu<sup>1</sup>, Gabriel Balan<sup>1</sup>, Mihai Boicu<sup>1</sup>, Gheorghe Tecuci<sup>1,2</sup>

<sup>1</sup>MSN 4A5, Learning Agents Laboratory, George Mason University, Fairfax, VA, 22030

<sup>2</sup>Center for Strategic Leadership, US Army War College, Carlisle, PA, 17013

{mbarbule, gbalan, mboicu, tecuci}@gmu.edu; http://lalab.gmu.edu

**Abstract** - *This paper presents the Disciple-RKF methodology for rapid development of large knowledge bases which relies on importing ontological knowledge from existing knowledge repositories, on parallel development of separate knowledge bases by subject matter experts, and on the merging of these knowledge bases into a high performance integrated knowledge base. The paper discusses several issues related to ontology import and merging, and presents the results of a successful knowledge base development and integration experiment performed at the US Army War College.*

**Keywords:** ontology import, knowledge bases integration, knowledge acquisition, intelligent agents, military center of gravity, experimental evaluation.

## 1 Introduction

In order to solve real-world problems, software agents need to incorporate complex models of the world, encoded into large knowledge bases. In the traditional approach to agent development the subject matter expert explains his reasoning to a knowledge engineer who formalizes it and encodes it into the agent's knowledge base. After that, the knowledge engineer and the subject matter expert analyze the way the agent reasons, find explanations for the agent's incomplete or incorrect solutions, and refine the knowledge base. This long, difficult and error-prone process is known as "the knowledge acquisition bottleneck" of agent development [2].

Recognizing the importance of advancing the knowledge base development technology, DARPA has sponsored two leading programs, High Performance Knowledge Bases, 1997-2000 [10] and Rapid Knowledge Formation, 2000-2004 [3]. The Learning Agents Laboratory at George Mason University has participated in both of them, continuing the development of the Disciple approach, an evolving theory, methodology and family of intelligent agent shells for rapid development of knowledge-based agents by subject matter experts, with limited assistance from knowledge engineers [18,19,20,21]. As a result, the current version of the Disciple approach allows rapid development of knowledge bases and agents for complex applications domains by

importing ontological knowledge from existing knowledge repositories, by enabling a team of subject matter experts that do not have prior knowledge engineering experience, to rapidly construct, update and extend knowledge bases in parallel, and by merging these knowledge bases into a high performance integrated knowledge base. This paper presents the general knowledge base development methodology of Disciple, particularly the ontology import and the knowledge base merging processes, supporting the claim of rapid development with the results of a knowledge base development and integration experiment performed at the US Army War College.

## 2 Methodology Overview

The Disciple approach relies on an instructable (learning) agent that can be taught directly by a subject matter expert to become a knowledge-based assistant. The expert teaches the agent how to perform problem solving tasks in a way that is similar to how the expert would teach a person. That is, the expert teaches the agent by providing it examples on how to solve specific problems, helping it to understand the solutions, and supervising and correcting the problem solving behavior of the agent. The agent, in turn, learns from the expert by generalizing the examples and building its knowledge base.

As shown in Figure 1, the rapid knowledge base development methodology of Disciple has four phases: domain analysis; building of an initial knowledge base; parallel development of knowledge bases for expertise sub-domains; and merging of the constructed knowledge bases into an integrated one.

The domain analysis phase is performed jointly by a knowledge engineer and a subject matter expert. During this phase they analyze typical problems in the application domain, informally outlining how the expert may solve these problems using the task reduction / solution synthesis paradigm. In this paradigm, a complex problem solving task is successively reduced to simpler sub-tasks, the solutions of the simplest sub-tasks are found, and these solutions are successively composed into the solution of the initial task. The Disciple methodology includes detailed guidelines and methods for helping the experts to express

their problem solving expertise using the task reduction paradigm. This approach was demonstrated in a variety of domains, including planning to repair damaged bridges, critiquing of military courses of action, and determining strategic centers of gravity in war scenarios [1].

There are three primary results of the “Domain analysis” phase: 1) an understanding by the subject matter experts of how to express their problem solving expertise using the task reduction paradigm, which they will use to teach personal Disciple agents; 2) an informal specification of the ontological terms (objects, relations, properties) needed in the knowledge base to be developed, which will guide the ontology import process; and 3) a partitioning of the application domain in different sub-domains for which knowledge bases will be developed in parallel, and then merged.

The second phase of the methodology consists in building an initial knowledge base consisting of an initial object ontology that represents the terms from a particular domain, and an initial set of problem solving (i.e. task reduction and solution composition) rules expressed with these terms. The object ontology is the more general component of the knowledge base, being characteristic to an entire domain, such as medicine, or military. A domain ontology specifies terms that are useful in a wide range of different applications in that domain. For instance, a military ontology would include specifications of military units and of military equipment that are very likely to be included in the knowledge base of any agent developed for a particular military application. Moreover, there is a wide agreement in any mature domain on the basic terms of that domain. This allows one to reuse ontological knowledge that was developed for previous applications in that domain. Therefore, Disciple includes modules for importing ontological knowledge from existing knowledge repositories, such as CYC [12]. Using the terms identified in the domain analysis phase as a guide, the knowledge engineer and the subject matter expert will search for formal descriptions of this ontological knowledge, and will import them into the object ontology under development. This ontology is further extended using the ontology browsers and editors of Disciple [16]. We describe the import method in detail in section 3.

The problem solving rules are a more specific component of the knowledge base. The rules are not only specific to a particular application in a given domain, but they are even specific to a particular subject matter expert. Consider, for instance, an agent that assists a military commander in critiquing courses of action with respect to the principles of war and the tenets of army operations [20]. The rules will identify strengths and weaknesses in a military course of action, and will obviously be domain specific. Moreover, they are very likely to include subjective elements that are based on the experience of a

specific military expert. Defining such problem solving rules is a very complex knowledge engineering task. In the Disciple approach, however, these rules are learned by a Disciple agent through a natural interaction with a subject matter expert, as described in [19, 21]. During the second phase of the methodology, the subject matter experts and the knowledge engineer teach a Disciple agent how to reduce the problems to solve into a set of sub-problems, where each sub-problem is a top-level problem in one of the sub-domains identified in phase 1 (see Figure 1).

In the third phase of the methodology, each subject matter expert receives a copy of the Disciple agent developed in phase 2, and is assigned one of the identified sub-domains of the domain. Then each subject matter expert will teach his/her personal Disciple agent how to solve problems in the corresponding sub-domain, and the agent will learn general problem solving rules and will extend the object ontology. The result will be n knowledge bases, one for each sub-domain of the initial domain. These n knowledge bases are then integrated into a single Disciple knowledge base. The new ontological terms defined by the experts are merged into a common ontology by using the method described in section 4. However, the rules learned by the different Disciple agents are kept in separate partitions of the integrated knowledge base. Therefore, when solving a complex problem, the final Disciple agent will first reduce it to a set of sub-problems, then each sub-problem will be solved in one rule partition, and the resulting solutions will be combined into a solution of the initial problem.

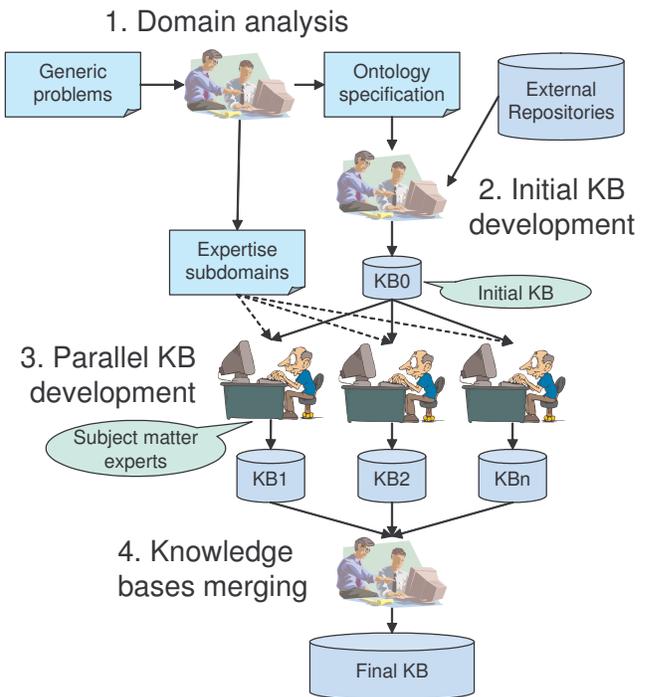


Figure 1. The phases of rapid knowledge base development

### 3 Ontology Import

In the initial KB development phase (see Figure 1), the knowledge engineer and the subject matter expert build an object ontology around the terms identified during the domain analysis phase. Instead of building it from scratch, they try to reuse knowledge already encoded in external repositories.

The ontology import process is depicted in Figure 2, and consists of three steps. First, there is an interactive step of identifying the objects in an external repository, such as CYC, that correspond to the terms specified during the domain analysis phase. We refer to these terms as the seed of the import process. Next, there is an automatic extraction of the knowledge related to these terms, into an Intermediate Ontology File. Finally, the extracted knowledge is translated and integrated into the Disciple ontology, through an interactive process.

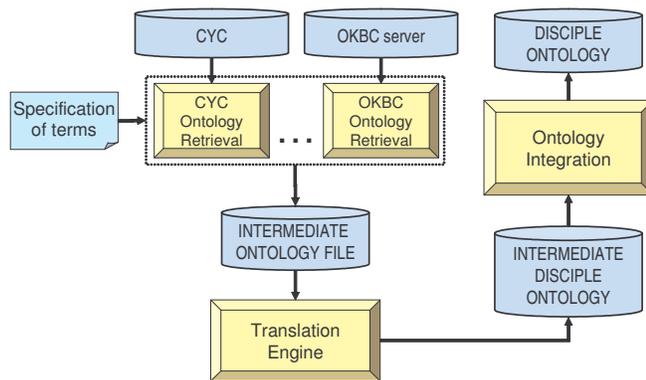


Figure 2. Ontology import architecture

An external repository is accessed through a dedicated module (e.g. CYC Ontology Retrieval, in Figure 2), which performs interactive seed discovery, automatic knowledge extraction and intermediate format translation. First, for every natural language term identified in the domain analysis phase, this module returns the objects from the external repository that are most likely to represent this term or a closely related one, and the expert has to select the relevant objects. So far, the only lexical relatedness metric we have used is substring matching, which is particularly useful since many frames have groups of words as names. However, synonyms of the original natural language term can also be used.

Once relevant terms from the external repository have been identified, the system performs an automatic extraction of other terms related to them, by using a slicing algorithm [6]. Intuitively, treating the ontology as a semantic graph, this algorithm computes the connected components of the nodes in the seed set. It produces a transitive closure of that set in terms of ontological

relationships. The slicing algorithm considers only those edges in the knowledge graph corresponding to axioms that can be accommodated by the destination knowledge representation. Because the original algorithm was designed with the slot-frame representation in mind, it uses the subclass-of, instance-of, domain-of, range-of and object-feature-value axioms. The algorithm follows the subclass-of and instance-of edges to get the generalizations of the objects in the seed. However, it does not follow these axioms “down” the generalization hierarchy. This could pose a limitation on the extracted knowledge. For example, the only way one could import a taxonomy of cars is to include the most specific car sub-concepts from the source repository into the seed. Special attention must be paid to these “downward” edges, since one could end up slicing (importing) the whole ontology regardless of the seed, thus defeating the whole purpose of the operation. Our slicing algorithm allows also to import generalization hierarchies of features.

The slicing algorithm performs a breadth-first search in the semantic graph. It also allows the user to specify a depth limit for the search. This is based on the observation that after a number of levels, some of the encountered terms tend to not longer be relevant to the original purpose of the seed. For instance, a catholic priest is a male person, which is conceptually related with male clothing, which is a subset of durable goods, which, in turn, is usually transported in trucks, trains or cargo ships.

The Translation Engine (see Figure 2) is responsible for the creation of an ontology consisting only of imported knowledge. It performs both a syntactic rewriting and a semantic one. The syntactic rewriting takes place during slicing and provides a one-to-one mapping from the extracted axioms to an intermediate representation which is an extension of Disciple’s knowledge representation.

We will describe the semantic rewriting with the most characteristic example of its use. In the Disciple knowledge representation, a frame cannot be both a class and an instance, such as in CYC. The system should decide whether to translate the imported frame into a concept, or to translate it into an instance, or to translate it both into an instance and into a concept. Such a decision can be made only after considering all taxonomic relationships the frame is involved in. However, the simpler model of translating a single axiom into another one is not powerful enough for this task. We say that local information is insufficient for rewriting one such taxonomic axiom, and global information is required. The semantic rewriting consists of building a temporary frame-based mirror of the sliced knowledge, and then querying it. While the assertional view and the frame based one are isomorphic, the later one is preferable when local information does not suffice. Notice that OntoMorph [4] uses a similar approach to the translation process.

We also use the distinction between the syntactic rewriting and the semantic rewriting for modularization purposes. Unlike the semantic stage, the syntactic one depends of the source repository. In order to import from other knowledge repositories, such as an OKBC server [5], one only needs to develop an appropriate wrapper of access functions and a specification of the rewriting of the extracted objects into the intermediate knowledge representation of Disciple. For the semantic rewriting, one needs to extend it only if the original design of the intermediate representation proves to be incomplete. In Klein's terminology [11], the language level mismatches have been taken care of. After this point, the knowledge has been transferred to Disciple, and the user will have to use only Disciple's knowledge representation and tools for the rest of the ontology import process. This last phase is performed by the Ontology Integration module (see Figure 2) which is described in the next section.

## 4 Knowledge Base Merging

Currently there is no automatic method of merging knowledge bases, all the existing approaches, including Prompt [15], Chimaera [13], and Disciple, use an interactive, semi-automatic approach.

We use knowledge base merging in two phases of our methodology. First we use it during the ontology import (phase 2 in Figure 1) for integrating the knowledge from multiple slicing operations into an initial ontology. Second, we use merging to integrate the knowledge bases that were developed in parallel during phase 3 (see Figure 1).

The knowledge needed in the ontology could be obtained through multiple extractions, from different external sources. The outcome of these extractions is a set of ontologies that must be combined into a single one. Our approach is to incorporate these ontologies one by one into an intermediary ontology, each time merging a new ontology into the intermediary ontology. This is similar to the approach described in [14], where the intermediary ontology plays the role of the "preferred ontology."

We have implemented a mixed-initiative ontology-merging tool by extending the functionality of the ontology editing tools of Disciple. The characteristic operation of this phase is the integration of a frame from one ontology into another. In the destination ontology, the copied frame can be coalesced into an existing one or a new frame can be created. If the whole frame is being transferred, this operation is called deep-copy. If only the name and the documentation of a frame are transferred, the action is called shallow-copy. The user is offered the possibility to shallow-copy a frame, deep-copy it or anything in between. A side effect of integrating a frame into the destination ontology is the creation of appropriate frames for all the names that were mentioned in the frame being copied, but

did not already exist. Because the created frames are incompletely defined, they are placed in a special list, with the purpose of reminding the user that they need to be fully specified. In this way the user is kept focused on the part of the ontology currently being integrated. The merging research efforts mentioned before have identified non atomic operations such as merge/remove concept, add/remove parent, etc. Our approach offers all these operations, but in a frame-oriented, more intuitive fashion.

One assumption in our approach is that terms with identical name in the source and destination ontologies should be semantically identical. Sometimes, when a frame from the source ontology is integrated into the destination ontology, its name is changed, either to match an existing frame or to follow a standardized naming pattern. This change is then automatically performed in the source ontology. This way, we maintain the link between the two frames in the two ontologies. Because the system relies heavily on name matching, consistent renaming is mandatory.

The developed merging capability is suitable for combining any pair of ontologies. However, the merging performed in phase 4 of our methodology (see Figure 1) has a particular flavor added to it. Because the ontologies to be merged originate from a common root (i.e. KB0) and share a common upper ontology, it makes sense for the user to be presented with a list of differences between them and with suggestions of what changes should be performed. We provide filters for the differences list, so the user can choose the appropriate level for the task at hand. These differences are at the semantic level, but they still rely on the previously stated assumption that same name implies same meaning. Throughout the merging process the user has to check for homonyms and coverage and granularity differences. This process is guided by heuristics such as the following one: if a name is a substring of another, the first might be a generalization of the second [13].

Using the ontologies merging operations as building blocks, we have also developed a module for merging the rules from the knowledge bases developed in parallel during phase 4, into a partitioned final knowledge base.

## 5 Rapid KB Development Experiments

In Spring 2002 a first rapid knowledge base development experiment was performed as part of the "Intelligent Agents" course at George Mason University. Each student had to develop an agent for helping someone choose a PhD advisor. The agent's goal was to identify strengths and weaknesses of potential advisors of the student. Six students trained personal Disciple agents in parallel, considering different characteristics of advisors and students. Then the six developed knowledge bases

were integrated into a final knowledge base. This experiment was conducted with students having prior knowledge engineering experience and it served as a dry run for an experiment with subject matter experts described in the following.

In Spring 2003 we have performed a unique experiment of parallel knowledge base development and merging using the methodology presented in Figure 1. This experiment was performed as part of the “Military Applications of Artificial Intelligence” course, at the US Army War College.

The goal was to have the students – high ranking military personal without extensive computer experience – to build an integrated knowledge base for determining the centers of gravity of the opposing forces in a war scenario. As defined by Carl Von Clausewitz in 1832 [7], the center of gravity of an entity (state, alliance, coalition, or group) is the foundation of capability, the hub of all power and movement, upon which everything depends, the point against which all the energies should be directed. If a combatant eliminates or influences the enemy’s strategic center of gravity, then the enemy will lose control of its power and resources and will eventually fall to defeat. If a combatant fails to adequately protect his own strategic center of gravity, he invites disaster [9].

The students have been provided with a Disciple agent previously trained to identify leaders as center of gravity candidates. The knowledge base of this agent consisted of 432 concepts and features, 29 tasks and 18 task reduction rules. We have then performed a joint domain analysis and ontology development with all the subject matter experts, by considering the example of testing whether Saddam Hussein, in the Iraq 2003 scenario, has all the required critical capabilities to be the center of gravity for Iraq 2003 [17]. In particular, we have analyzed whether Saddam Hussein has the capabilities to be protected, stay informed, communicate, be influential, have support, be a driving force, and be irreplaceable. Then we have identified which are the critical requirements for these capabilities to be operational, and which are their critical vulnerabilities. Based on this analysis, the Disciple’s ontology was extended by a knowledge engineer with 37 new concepts and features identified as relevant by the subject matter experts.

The 13 subject matter experts have then been grouped into five teams (of 2 or 3 experts each), and each team was given a copy of the extended Disciple agent. Each team has trained its agent to test whether a leader has one or two of the critical capabilities mentioned above (e.g. the capability to be protected). The training was done based on three scenarios (the Iraq 2003 war, the Arab-Israeli 1973 war, and the current war on terror), the experts teaching Disciple how to test each strategic leader from that scenario. As a

result of the training performed by the subject matter experts, the knowledge base of each Disciple agent was extended with learned features, tasks, and rules. The average training time for each team was 5.47 hours. The teams have learned a total of 99 rules, with an average rate of 3.53 rules/hour, and from an average of 2.5 examples/rule. This supports our claim of rapid knowledge base development.

Then the knowledge bases of the five Disciple agents have been merged by a knowledge engineer. During this process two semantically equivalent features have been unified, 4 incomplete rules were deleted, and 11 other rules were refined.

Next, each team has tested the integrated agent on a new scenario and has been asked to judge the correctness of the solutions generated by the agent (but only with respect to the capabilities for which that team performed the training of the agent). The result was 98.15% correctness, which supports our claim of developing high performance knowledge bases.

To our knowledge, this is the first time that such an experiment has been performed. It demonstrates Disciple’s capability for rapid development of high performance knowledge bases by subject matter experts, with limited assistance from knowledge engineers.

## 6 Conclusions and Future Work

We have presented a methodology for rapid development of large knowledge bases, by a team of subject matter experts. We have shown how to speed up the development process by reusing previously developed knowledge and by developing different parts of the knowledge base in parallel. We have discussed in more detail issues related to the implementation of translation and merging of ontologies. We have also presented several experiments that validate our methodology.

We plan to extend our current work in several directions: refine the import algorithm in order to limit the slicing of irrelevant data; develop wrappers to allow importing from other external knowledge sources such as OKBC knowledge servers [5] or DAML+OIL ontologies [8]; improve the pro-activity of the mixed-initiative ontology merging assistant.

**Acknowledgments.** This research was sponsored by DARPA, AFRL, AFMC, USAF, under agreement number F30602-00-2-0546, by the AFOSR under grant no. F49620-00-1-0072, and by the US Army War College. Several members of the LALAB, particularly Dorin Marcu, Bogdan Stanescu, and Cristina Boicu, contributed to the development of Disciple-RKF.

## References

- [1] Bowman, M., *A Methodology for Modeling Expert Knowledge that Supports Teaching Based Development of Agents*, Ph.D. Dissertation, George Mason Univ, 2002.
- [2] Buchanan, B. G. and Wilkins, D. C. eds. *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*. San Francisco, CA: Morgan Kaufmann, 1993.
- [3] Burke, M., *Rapid Knowledge Formation Program Description*, at [http://reliant.teknowledge.com/RKF/projects/Darpa\\_RKF\\_PIP.htm](http://reliant.teknowledge.com/RKF/projects/Darpa_RKF_PIP.htm), 1999, accessed on 3 July 2003
- [4] Chalupsky, H., "OntoMorph: A Translation System for Symbolic Knowledge", In A.G. Cohn, F. Giunchiglia, and B. Selman, eds, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference*. San Francisco, CA: Morgan Kaufmann, 2000.
- [5] Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D. and Rice, J. P., "OKBC: A Programmatic Foundation for Knowledge Base Interoperability", *Proc. of the 15th National Conference on Artificial Intelligence*, 600-607, Madison, Wisconsin: AAAI Press/The MIT Press, 1998.
- [6] Chaudhri, V. K., Stickel, M. E., Thomere, J. F. and Waldinger, R. J., "Using Prior Knowledge: Problems and Solutions", *Proc. of the 17th National Conference on Artificial Intelligence and the 12th Conference on Innovative Applications of Artificial Intelligence*, 436-442. Austin, Texas: AAAI Press/The MIT Press, 2000.
- [7] Clausewitz, C.V. 1832, *On War*, translated and edited by M. Howard and P. Paret. Princeton, NJ: Princeton University Press, 1976.
- [8] Connolly, D., van Hermelan, F., Horrocks, I., McGuinness, D., Patel-Schneider, P. F., and Stein, L. A., "Reference Description of the DAML+OIL Ontology Markup Language", W3C Note 18 December 2001.
- [9] Giles, P.K., and Galvin, T.P., *Center of Gravity: Determination, Analysis and Application*, CSL, U.S. Army War College, PA: Carlisle Barracks, 1996.
- [10] Gunning, D., *High Performance Knowledge Bases Program Description*, at <http://reliant.teknowledge.com/HPKB/about/about.html>, 1996, accessed on 3 July 2003.
- [11] Klein, M., "Combining and Relating Ontologies: An Analysis of Problems and Solutions", *Proceedings of the IJCAI-2000 Workshop on Ontologies and Information Sharing*, Seattle, Washington: IJCAI, Inc., 2000.
- [12] Lenat, D. B., "CYC: A Large-scale Investment in Knowledge Infrastructure", *Communications of the ACM*, 38(11): 33-38, 1995.
- [13] McGuinness, D. L., Fikes, R., Rice, J., and Wilder, S., "An Environment for Merging and Testing Large Ontologies", *Proc. of the 7th International Conference on Principles of Knowledge Representation and Reasoning*, 483-493, Breckenridge, CO: Morgan Kaufmann, 2000.
- [14] Noy, N. F. and Musen, M. A., "An Algorithm for Merging and Aligning Ontologies: Automation and Tool Support", *Proc. of the AAAI-99 Workshop on Ontology Management*, Orlando, Florida: AAAI Press, 1999.
- [15] Noy, N. F. and Musen, M. A., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment", *Proc. of the 17th National Conference on Artificial Intelligence*, Austin, Texas: AAAI Press, 2000.
- [16] Stanescu, B., Boicu, C., Balan, G., Barbulescu, M., Boicu, M., Tecuci, G., "Ontologies for Learning Agents: Problems, Solutions and Directions", *Proc. of the IJCAI-03 Workshop on Ontologies and Distributed Systems*, Acapulco, Mexico, August 2003.
- [17] Strange, J., *Centers of Gravity & Critical Vulnerabilities: Building on the Clausewitzian Foundation So That We Can All Speak the Same Language*, Quantico, VA: Marine Corps University Foundation, 1996.
- [18] Tecuci G., *Disciple: A Theory, Methodology and System for Learning Expert Knowledge*, Thèse de Docteur en Science, University of Paris-South, France, 1988.
- [19] Tecuci G., *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*, San Diego: Academic Press, 1998.
- [20] Tecuci G., Boicu M., Bowman M., and Marcu D., with a commentary by Burke M., "An Innovative Application from the DARPA Knowledge Bases Programs: Rapid Development of a High Performance Knowledge Base for Course of Action Critiquing", *AI Magazine*, 22, 2, pp.43-61, AAAI Press, Menlo Park, CA, 2001.
- [21] Tecuci G., Boicu M., Marcu D., Stanescu B., Boicu C., Comello J., "Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain", *AI Magazine*, 24, 4, pp.51-68, AAAI Press, Menlo Park, CA, 2002.