In Proceedings of ICCM - 2007- Eighth International Conference on Cognitive Modeling. 325 - 330. Oxford, UK: Taylor & Francis/Psychology Press.

Towards a Complete, Multi-level Cognitive Architecture

Robert Wray[1] **(wray@soartech.com), Christian Lebiere**[2] **(cl@cmu.edu), Peter Weinstein**[3] **(peter.weinstein@altarum.org), Krishna Jha**[4] **(kjha@atl.lmco.com), Jonathan Springer**[5] **(springer@reservoir.com), Ted Belding**[6] **(ted.belding@newvectors.net), Bradley Best**[7] **(bradjbest@gmail.com) & Van Parunak**[6] **(van.parunak@newvectors.net)**

[1]Soar Technology, Inc., 3600 Green Court Suite 600, Ann Arbor, MI 48105
[2]Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213
[3]Altarum Institute, 3520 Green Court Suite 300, Ann Arbor, MI 48105
[4]Lockheed Martin Advanced Technology Labs, 3 Executive Campus, Cherry Hill, NJ 08002
[5]Reservoir Labs, 632 Broadway Suite 803, New York, NY 10012
[6]NewVectors, 3520 Green Court Suite 250, Ann Arbor, MI 48105
[7]Adaptive Cognitive Systems LLC, 1709 Alpine Avenue, Boulder, CO 80304

Abstract

The paper describes a novel approach to cognitive architecture exploration in which multiple cognitive architectures are integrated in their entirety. The goal is to increase significantly the application breadth and utility of cognitive architectures generally. The resulting architecture favors a breadth-first rather than depth-first approach to cognitive modeling by focusing on matching the broad power of human cognition rather than any specific data set. It uses human cognition as a functional blueprint for meeting the requirements for general intelligence. For example, a chief design principle is inspired by the power of human perception and memory to reduce the effective complexity of problem solving. Such complexity reduction is reflected in an emphasis on integrating subsymbolic and statistical mechanisms with symbolic ones. The architecture realizes a "cognitive pyramid" in which the scale and complexity of a problem is successively reduced via three computational layers: Proto-cognition (information filtering and clustering), Micro-cognition (memory retrieval modulated by expertise) and Macro-cognition (knowledge-based reasoning). The consequence of this design is that knowledge-based reasoning is used primarily for non-routine, novel situations; more familiar situations are handled by experience-based memory retrieval. Filtering and clustering improve overall scalability by reducing the elements to be considered by higher levels. The paper describes the design of the architecture, two prototype explorations, and evaluation and limitations.

Introduction

Cognitive architecture research seeks to define a collection of integrated processes and knowledge representations that provide a general foundation for intelligent behavior across an increasingly broad spectrum of problems. While most cognitive architectures seek to cover the requirements for general intelligence, all today fall far short of that goal. They represent "works in progress" that tend to have niche strengths in a few problem domains.

Investigations of human psychology suggest that human behavior derives from the integration of three mechanisms: 1) powerful perceptual and "pre-cognitive" processing; 2) memory, along with processes that contextualize and generalize experience to make it readily applicable in future situations; and 3) satisficing ("good enough") reasoning. As an example, Best (2005) found that humans generate reasonably good solutions to the NP-complete traveling salesman problem (TSP) in nearly linear time for problems up to about 40 cities. Humans accomplish this via parallel perceptual processes that group cities by proximity, evaluation of goodness of path (dependent on experience and memory), and a localized, serial search process. We introduce C3I1 ("three cognitions, one intelligence.") a novel cognitive architecture that integrates three cognitive-architecture-level approaches, each targeted to one of these distinguishing human capacities.

Our approach is similar to the common approach to cognitive architecture development, which adapts and incorporates existing algorithms into an architecture. Cassimatis (2006) and Chong and Wray (2005) offer recent examples of the typical approach. Our approach is different primarily in terms of scale: rather than integrating individual algorithms, we map three cognitive-architecture-level systems to each of the three functionalities outlined above. The methodological hypothesis motivating this approach is that by integrating these complete, relatively mature approaches, we can enable faster exploration of alternatives in design space than approaches that integrate sub-components of existing cognitive architectures. If true, this will enable more rapid and thorough evaluation of the three-level architecture than constructing *de novo* a new architecture designed around the functional capabilities.

An obvious potential drawback is the potential redundancy in functional capability at each level, with consequences for software engineering complexity and run-time performance. The architecture reported in this paper is not the end-goal, but rather a description of the results of our experiences along this methodological path. The results suggest that even the superficial integrations we report here have significant functional value. However, long-term, our goal is to refine and deepen the initial integrations described here, based on lessons learned from the explorations.

This paper reports the initial progress and lessons learned in terms of evaluating the methodological hypothesis. We report progress toward three goals: 1) defining the functional requirements for each level of the architecture, 2) evaluating the utility of the general design pattern via prototype implementations, and 3) identifying functional synergies that could result long-term in tighter integration. In order to explore design consequences empirically, we have chosen specific candidate systems for each level of the system. Pre-cognitive processing, or Proto-cognition, is accomplished via configurations of swarming algorithms. ACT-R is used for memory formation and expertise-based retrieval (Micro-cognition), serving as a bridge between the sub-symbolic Proto-cognitive component and a symbolic Macro-cognitive layer. Macro-cognition, the locus of symbolic reasoning and explicit knowledge, is realized via Soar. Each instantiation was chosen because of its maturity and capability with respect to the practical realization of the target functionality, rather than any regard to cognitive plausibility or a definite commitment to a particular choice of method for any level. Other approaches to the same functionality could be substituted, allowing for an evaluation of the properties of constituent mechanisms.

Design Principles

The design of C3I1 is inspired by two principles particularly important in the human cognitive architecture: progressive filtering and structuring of perception ("cognitive pyramid") and tight integration of functionalities and modalities.

The Cognitive Pyramid

There is a computational trade-off between the quantity of data that can be processed at any one time and the sophistication of reasoning applied to that data. The human cognitive architecture solves this problem by collapsing and integrating large volumes of perceptual information into more abstract aggregates that can then be stored, retrieved, manipulated and composed more tractably. We refer to this volume-complexity tradeoff as a "cognitive pyramid". When the volume of incoming data is high, the architecture applies local processing to the data, the function of which is to aggregate and filter. At progressively higher levels in the architecture, because total data volume is reduced, more computationally expensive methods (such as symbolic reasoning) can be brought to bear tractably.

While this principle has been laid out in purely computational terms, the architecture of the human brain follows similar principles for information processing. When information reaches the brain, it is first processed by the sensory cortex. Subsequent rapid processing greatly reduces the complexity of the data into increasingly abstract representations. Quick reaction can be obtained by comparing that abstracted information to stored patterns in massively parallel procedural and declarative information stores hypothesized to be located in the basal ganglia and neocortex, respectively. If more sophisticated processing is required such as reasoning or planning, relatively slow, sequential processing occurs, controlled by contextual structures in the prefrontal cortex. While we do not equate the three cognitive levels with specific brain structures, nor argue for the neural plausibility of our algorithms, the organizing principles bear substantial similarities.

Tight Integration

A second architectural principle for C3I1 is tight integration: fine-grained, full-spectrum interaction between the cognitive levels. Tight integration may seem like an unintuitive requirement, given the components to be integrated. However, both Soar and ACT-R suggest that tight integration results in better leveraging of the inherent power of individual components (Anderson et al. 2004; Jones and Wray 2006). With loose integration, information about the situation becomes "trapped" inside subsystems and cannot be efficiently communicated. In a tightly integrated approach, all cognitive components will share a common memory, integrated control process, and a shared language for communication with individual components.

The neurological architecture of human cognition is characterized by both forward and reverse projections from interconnected areas. This observation inspired a design choice that each information flow between cognitive layers would have a reverse learning flow. For example, even in the prototype we describe, Proto-cognition's filtering and clustering are modulated by signals from Micro-cognition indicating goodness-of-fit and applicability of high salience clusters to on-going problem solving. This allows the architecture to adapt and optimize itself to the nature of its processing and the structure of its environment. In this paper, we focus on describing the implemented and tested version of the architecture. This prototype currently falls far short of a tightly integrated architecture, but suggests specific opportunities for beneficial, synergistic integrations as discussed in the conclusions.

C3I1 Implementation

This section outlines the role of the three primary components of the architecture. Importantly, the individual architectures proposed for each level are not unique choices; other architectures or methods likely could have been applied at each level. The focus instead concerns how these more general systems have been applied to the specific role proposed for each level in the architecture.

Proto-cognition

The primary role of Proto-cognition ("Proto") in the architecture is to reduce the scale of raw input data. This general goal may be accomplished in a variety of ways; the current design envisions two functions: 1) organizing data into structured, hierarchical representations and 2) providing skeletal subsolutions for the higher layers. These processes reduce the effective search space for other layers. Currently, we have explored two specific swarming-based mechanisms for Proto: SODAS (decentralized hierarchical clustering) and marker-based stigmergy (topological sorting).

SODAS (Parunak et al. 2006) implements dynamic decentralized hierarchical clustering via swarming. In SODAS, nodes represent either data nodes or summaries of subtrees of nodes; links represent parent-child relationships, so that the network forms a hierarchical tree of data clusters. SODAS' merge operator is identical to classical agglomerative hierarchical clustering, providing a fully general hierarchical clustering method. SODAS' promote operator provides SODAS with the additional ability to reorganize and improve existing cluster trees, making it arguably more general than most other clustering methods.

Marker-based stigmergy depends on special-purpose information ("pheromones") that agents deposit in the environment to support coordination. An example in nature is ant foraging, where ants that have found food leave pheromones as they return to the nest, which attract other ants to the food source. In the context of C3I1, it is used to provide skeletal solution frameworks to the higher layers for such problems as topological sorting. Here, the nodes are spatial locations, place agents, and each node has links to each of its spatial neighbors (for instance in a 2D lattice). Swarming agents roam over this spatial topology, reacting to pheromone concentrations and other agents, and depositing their own pheromones.

Micro-cognition

The role of Micro-cognition ("Micro") is to store and deploy problem-solving expertise. A key part of the ability to solve complex problems efficiently is in learning from experience and then deploying that knowledge as a substitute for the time-intensive reasoning processes by which it was accumulated. One can view the process as a space-time tradeoff, where the system trades increased storage space for expertise in exchange for reduced processing time in finding an expert-level solution. This approach relies on the engineering assumption that storage is increasingly less expensive and that special-purpose hardware can provide storage with the right properties (robust associative content-based retrieval, constant time access, etc.); response time is usually a less negotiable constraint. The C3I1 prototype uses ACT-R (Anderson and Lebiere 1998) for the Micro-cognitive layer. ACT-R includes functionality for generalizing patterns that take the form of either declarative knowledge, capturing a solution to various sub-problems, or procedural skills, resulting in more efficient execution.

Micro is defined by sequential cognition that relies on a hybrid mix of symbolic and statistical knowledge, such as applying a set of patterns to the current situation to make the best guess under tight computational constraints. The computation is a mix of sequential and parallel, similarity- and history-based processes and bottom-up and top-down constraints. Key operations include retrieving declarative knowledge from long-term memory and selecting the most promising rule from the procedural skill set. These operations are implemented by a combination of symbolic pattern-matching and subsymbolic processes:

Partial matching compares stored memories to the current problem to determine the closest-matching pattern. A current problem will seldom exactly match previous experience. Partial matching uses similarity-based generalization similar to that of neural networks to find the closest match to an item in memory. While the pattern is described symbolically, the internal mechanism generalizing that pattern is statistical.

Bayesian learning operates at the subsymbolic level of declarative memory. Its role is to guide the retrieval process to the most likely pieces of relevant information. It combines parameters derived from computations at the Proto level that reflect factors such as the compatibility between problem and solution, together .with quantities learned at the Micro level that capture environmental heuristics such as recency and frequency of use.

Reinforcement Learning controls and tunes procedural skill acquisition, using successes and failures as input. Subsymbolic utilities are associated with skills to control selection and application and learned using online reinforcement learning algorithms. RL alone typically scales poorly in combinatorial state spaces, but using RL with general production rules instead of discrete, combinatorial states significantly improves efficiency.

Macro-cognition

The primary task of Macro-cognition ("Macro") in C3I1 is to perform a deliberate "problem search" to solve a specific (sub)problem initiated by Micro. In essence, Macro brings knowledge to bear in deliberating over non-routine, novel tasks. Soar (Newell, 1990) is used for Macro-cognition.

Problem search is potentially expensive. In order to reduce the cost of problem search, Macro should ensure responsiveness to the current problem (not waste cycles working on an out-dated problem), support multiple methods of reasoning (apply the most appropriate reasoning to the problem), support efficient knowledge search (minimize the cost of bringing knowledge to bear) and improve performance with experience. Soar implements and integrates a number of influential ideas and algorithms that support these requirements, including:

Efficient, pattern-directed control: The flow of control in Macro is determined by context and the associations it triggers. Macro requires least commitment representation; there should be no fixed, design-time mapping between the justifications for an activity and its implementation. This distinction supports multimethod reasoning because the approach to some specific activity (e.g., resolving a request from Micro) can be pursued in multiple different ways.

Reason maintenance: Reason maintenance ensures that Macro is responsive to the environment. It embeds dynamics of belief change directly in the architecture. This facilitates multimethod reasoning because reason maintenance is implemented consistently across any reasoning method.

Meta-reasoning: Macro can recognize when it has no available options or has conflicting information about its options and, in response, creates a goal to resolve this "impasse." This process facilitates multimethod reasoning, because different approaches (e.g., reasoning by analogy vs. lookahead planning) can be identified and brought to bear to attempt to resolve individual subgoals.

Knowledge compilation: Macro should improve with experience, converting the results of reasoning to new knowledge representations that summarize reasoning. This allows the system to skip reasoning steps when a similar situation is later encountered, improving responsiveness.

Memory and Control

In addition to the individual components, the architecture requires memory store(s), a control process, and a message passing language and infrastructure.

Unified Memory A significant design decision was whether to create a single memory, shared by each of the three layers, or to use the individual memory systems available within each of the existing components. A unified memory would be more efficient than passing partial results, especially given high volumes of data (e.g., Proto passing up clustering results). A shared memory also ensures consistency by making sure that all cognitions work from the same problem state. While, in theory, separate memories for each cognition could be synchronized, in practice, resolving synchronization issues (such as race conditions) is oftentimes difficult and labor-intensive. On the other hand, the three cognitions currently have significantly different memory models. A common memory model might be inefficient for the operations of individual cognitions. For instance, memory organized for Soar and ACT-R might not be a practical organization for use by swarming. As well, reimplementation costs are significant. While the various cognitions are somewhat modular, changes to their memory systems would have broad repercussions throughout their implementations. We used the distinct memories of the individual cognitions in the prototype, but are also pursuing unified memory integration.

Integrated Control Explicit control is needed to coordinate the three cognitions. Integrated control would mean a single process that would invoke the primitives provided by the three cognitions. Integrated control would also enable it to be more adaptive and generally more sensitive to the operations of the cognitions than control structures located within each cognition or outside of the cognitions entirely. The prototype assumes a simple up-and-down data flow: Proto computes a first pass, Micro attempts to apply its knowledge to some subproblem, and Macro is called upon if Micro's expertise is insufficient for the problem. Results then flow down to Micro and Proto and the cycle repeats. In parallel, there are reverse flows for each information flow, providing modulation and feedback. We also developed explicit control strategies for each prototype application. This approach is not an appropriate long-term solution, but it allowed an exploration of the requirements and implications of integrated control.

Inter-cognition Communication & Control: In order to communicate with other components, and with external systems that present "problems" to the architecture, we developed a XML-schema-based language. Message types are organized into three categories:

Knowledge management. Messages (e.g., `AddChunk`, `RemoveChunk`) allow management of problem data and general domain knowledge. For example, any of the cognitions could send an `AddChunk` message when a new inference or cluster was created, which would then be communicated to the other cognitions.

Cognition invocations & responses. Messages such as `MicroRequestInference` encode requests between cognitive components and corresponding responses (e.g., `MacroRespondInference`).

Operational protocols. Protocols specify how to declare problems to C3I1 and how to deliver solutions.

The set of messages is fixed and general across problem domains. We developed functions within Swarming, ACT-R, and Soar to interpret the messages in a domain-general manner. The types of data contained in the messages are problem-specific. For example, the AddChunk message is always used to add data, but the structure of the data added will be different for different applications. Thus the language is modular; there is a generic part into which domain-specific parts can be inserted. XMLBlaster is used to transmit messages between cognitions, facilitating a distributed, flexible experimentation environment.

Prototype Explorations

In order to elicit requirements and empirically explore implications of C3I1, we built prototypes for two very different application domains: evidence marshalling and unmanned-aerial vehicle (UAV) route planning.

Evidence Marshalling

Evidence marshalling is the process of collecting and organizing information (evidence) to support a hypothesis. We used a simple but realistic data set used to train intelligence analysts. In this scenario, many distinct reports provide hints about a possible coordinated terrorist attack in three different cities. We applied C3I1 to uncovering specific connections in this dataset (after hand-encoding facts from the reports into C3I1 assertions). Proto-cognition uses SODAS to cluster similar facts together. Micro applies its expertise to "link" facts and clusters and infer new conclusions. When it lacks knowledge to make a link, Micro calls Macro. Macro, using ontologies and domain knowledge, searches for a link between facts of interest.

The primary lesson learned from this prototype was that the basic functional roles we had defined in the design demonstrated utility in application. It also allowed us to define and implement the initial version of the interaction language and give the cognitions the ability to pass messages. However, we also quickly recognized that evidence marshalling was a poor prototyping domain: there would be significant expense to encode a very large database of facts, but a large store of facts was needed to demonstrate the power of Proto's clustering. The example problem was small enough (< 1000 assertions) that the problem could be readily addressed within ACT-R or Soar.

Route Planning for Unmanned Aerial Vehicles

The application, a simplified representation of the routing of an unmanned aerial vehicle (UAV), includes these elements:

• **Terrain**: The scenario was defined to take place on a 2D terrain overlaid with a 100x100 coordinate grid. Positions of other components were specified in terms of this grid.

• **Targets**: A set of target locations on the map. The UAV had to visit all targets, in an order of its choosing.

• **Threats**: A set of threat locations was defined on the map, each with a threat radius. The UAV had to complete its tour without entering any threat radius.

We defined this scenario to be very similar to the Traveling Salesman Problem (TSP). However, threats and dynamic ("pop-up") locations were included in the scenarios, which give the problem more realistic requirements. Figure 1 illustrates the basic solution developed, drawing heavily from the aforementioned model of Best (2005). Proto both clusters target points (in 1a) via SODAS and produces, via marker-based stigmergy, a skeletal solution (in 1b) representing possible routes between targets in a particular cluster. Although not shown in the figure, Proto performs these functions in a space warped by threats, so that simple Euclidean distance is not sufficient for producing "good" potential routes.

While Proto continues to cluster targets (and clusters of targets), Micro attempts to match groups of clusters against prior routes stored in memory. Initially, it will have no matching routes; Macro is invoked to find a routing between clusters. Because the number of clusters at the highest level is small, the routing was determined via a straightforward implementation of a TSP solution (in 1c), using distance measures between targets as computed by Proto (in 1b). We also explored the use of heuristic planning within Macro, using categorizations of rough route geometries (arcs, lines, and loops, e.g.,) to simplify route planning. However, in practice, Proto reduced the number of targets sufficiently that Macro could easily compute an analytic solution.

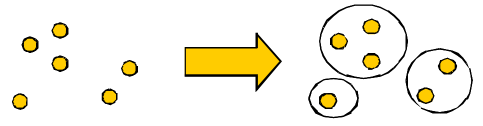
Once Macro finds a routing, it provides the route to Micro, which stores the route in memory. As Micro gains experience, it will increasingly be able to retrieve route segments from its memory (in 1d), without recourse to Macro. A hardware support strategy has been developed for this memory, based on an understanding of the functionality of the human visual system and strategies that are used for visual search (Pederson et al. 2006).

Once a route is computed between clusters, the system recursively develops routes within the targets (and/or subclusters) within each of the clusters in the evolving route. This approach also has the desired property that pop-up targets and pop-up threats typically introduce local changes, requiring only new routing within a cluster, rather than a high-level reconsideration of the overall route. We did explore tunings of SODAS clustering so that it would prefer making minimal changes to existing clusters when dynamic locations or threats were presented.

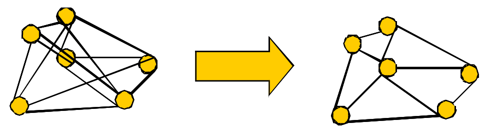
The approach we have taken does not guarantee a solution in linear time but is, on average, much less expensive than an analytic solution. The routes resulting from this decomposition are obviously also not always optimal but are generally close to it. We have observed some cases in which poor routes are generated and are investigating using Macro as a "critic" to recognize these situations and repair them. This additional functionality may represent a meta-cognitive role for Macro-cognition in the future.

Evaluation & Conclusions

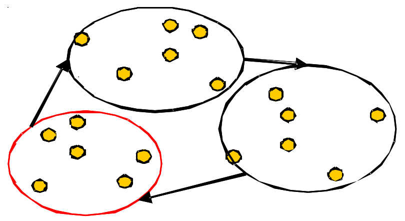
We proposed C3I1 to address a greater range and scale of problems with an architecture. We explored two different applications domains to evaluate the extent to which these goals are being achieved. We consider three general



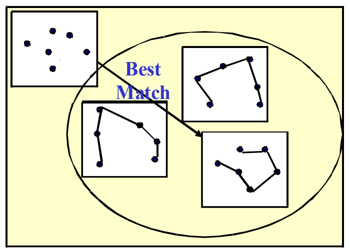
(a) Proto, via SODAS, hierarchically clusters target locations, modulated by threat locations.



(b) Proto, via marker-based stigmergy, creates plausible routes between nodes and clusters.



(c) Macro, via a simple planning algorithm, chooses orderings of clusters and nodes.



(d) With experience, Micro is able to generalize previous routes to current problems without invoking Macro.

Figure 1: Operation of C3I1 on route planning example

dimensions of evaluation: generality, taskability, and efficiency. While these criteria appear purely functional, they are similar to proposed tests for a theory of cognition (Anderson & Lebiere, 2003) such as behaving arbitrarily as a function of the environment and operating in real time.

Generality: C3I1 seeks to provide a very general computational cognitive substrate, enabling both high degrees of applicability and robustness. As a general architecture, C3I1 shows promise. The same core mechanisms and design demonstrated value in the highly symbolic, knowledge-intensive evidence marshalling domain, as well as in the more algorithmic, more dynamic routing problem. However, special-purpose mechanisms were developed within each layer for each application (e.g., in routing, threat-warped clustering, topologically-oriented memory retrieval, and the TSP sorting heuristics). Our investigations, however, suggest a fixed, domain-independent set of mechanisms for Proto is feasible.

Taskability: C3I1 should not only apply to a broad range of problems, but it should do so without requiring extensive developer modification and have inherent ability to adapt to the requirements and constraints of particular problems. Taskability includes the ability to handle novel variations of problems and improve with experience. It also includes extensibility: the architecture should require small (and increasingly smaller) human effort to achieve good results in different applications. The taskability of C3I1 is both less clearly encouraging and more tentative. In the near-term, significant developer involvement will be needed in each layer; essentially, the approach requires a manual decomposition and mapping of the problem to the functionality of the three layers. Over time, we believe that tighter integration of the mechanisms will improve taskability, as a more constrained architecture will guide solutions more explicitly.

Efficiency: General, architectural solutions are always likely to be less efficient than special-purpose ones. To compensate, overall efficiency of C3I1, especially in the core loops of processing, is critical. Efficiency includes responsiveness to specific situations, overall resource management, and scalability across problems of increasing complexity. We have evaluated the computational bounds at each level. While presentation of this analysis is not possible in this brief paper, in terms of worst-case performance, the architecture should perform acceptably well, assuming specialized, parallel hardware for Proto and Micro. A key assumption is that Proto will reduce the overall scale of the problem by several orders of magnitude, in order to make Macro problem search generally tractable. Our initial explorations suggest Proto can accomplish a level filtering and aggregation that greatly reduces the raw scale of reasoning problems, but it is an open, empirical question how generally this power can be realized across domains and applications.

In terms of the methodological hypothesis, the explorations did identify potential opportunities for much tighter, more synergistic integrations of the components. Integration between swarming and ACT-R will focus on unifying the ACT-R activation calculus and the swarming output, including devising methods for learning within ACT-R to influence Swarming. Integration between ACT-R and Soar based on the pattern of usage in the prototypes would center on impasses and chunking. ACT-R currently calls upon Soar when no relevant knowledge exists to apply directly, which is a concept very similar to the Soar impasse mechanism, but not a process directly supported in ACT-R. The information returned by Soar is a summary of the result of its reasoning. Thus, this process is functionally identical to Soar's chunking and makes it a good candidate for the focus of integration for Soar and ACT-R.

As mentioned, other alternatives are possible for the cognitions of C3I1. For example, neurologically-inspired clustering might be appropriate for perceptually-dominated domains. These observations point to the question of whether C3I1 is an architecture, making specific commitments to particular mechanisms, or a framework, representing a design pattern for building general, intelligent systems but not making specific commitments to representations and algorithms. While our intention is to follow up the promise observed in the prototypes by investigating more fine-grained integration of the current technologies, it would also be equally valid to explore and evaluate alternative technologies. Similarly, the C3I1 decomposition also serves as a suggestive guide for developing less brittle and more scalable solutions to specific classes of domain problems (such as route planning), where the goal is a point solution. All of these options point to the value of considering existing cognitive architectures as computational primitives for integration into larger and increasingly capable intelligent systems.

Acknowledgments

The research reported in this paper was performed via the support of the DARPA Architectures for Cognition Information Processing (ACIP) program, contract FA8750-04-C-0266, administered by the US Air Force Research Laboratory. The authors would like to thank program managers Robert Graybill and William Harrod for the support of this research. DARPA Distribution Statement A (Approved for Public Release, Distribution Unlimited), DISTAR Case #8726, 3/28/2007.

References

Anderson, J. R., and Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111(4)*, 1036-1060.

Anderson, J. R. & Lebiere, C. (2003). The Newell test for a theory of cognition. Beh. & Brain Sciences 26, 587-637.

Best, B. J. (2005) A Model of Fast Human Performance on a Computationally Hard Problem. *Proceedings of the 27th Annual Conference of the Cognitive Science Society*.

Brueckner, S. A., and Parunak, H. V. D. (2002) Swarming Agents for Distributed Pattern Detection and Classification. *AAMAS Workshop on Ubiquitous Computing*. Bologna, Italy.

Cassimatis, N. L. (2006). A Cognitive Substrate for Human-Level Intelligence. *AI Magazine*, 27(2).

Chong, R. S., and Wray, R. E. (2005). Constraints on Architectural Models: Elements of ACT-R, Soar and EPIC in Human Learning and Performance. In *Modeling Human Behavior with Integrated Cognitive Architectures*, Gluck, K. and Pew R., eds., Erlbaum, 237-304.

Jones, R. M., and Wray, R. E. (2006). Comparative Analysis of Frameworks for Knowledge-Intensive Intelligent Agents. *AI Magazine*, 27, 57-70.

Newell, A. (1990). *Unified Theories of Cognition.* Cambridge, MA: Harvard University Press.

Parunak, H. V. D., Rohwer, R., Belding, T. C., and Brueckner, S. A. (2006) Dynamic Decentralized Any-Time Hierarchical Clustering. *Fourth International Workshop on Engineering Self-Organizing Systems.*

Pederson, K., Lethin, R., Springer, J., Manohar, R., and Melhem, R. (2006) Enabling Cognitive Architectures for UAV Mission Planning. *10th Annual Workshop High Performance Embedded Computing* (HPEC 2006).