**19th ICCRTS**

***C2 Agility: Lessons Learned from Research and Operations***

**Paper ID: 073**

# Towards a Cognitively Realistic Computational Model of Team Problem Solving Using ACT-R Agents and the ELICIT Experimentation Framework

| | |
|---|---|
| **Primary Topic:** | Topic 5 – Modeling and Simulation |
| **Alternative Topics:** | Topic 2 – Organizational Concepts and Approaches |
| | Topic 4 – Experimentation, Metrics and Analysis |

**Authors**

**Paul R Smart**
University of Southampton, Southampton, UK

**Darren P. Richardson**
University of Southampton, Southampton, UK

**Katia Sycara**
Carnegie Mellon University, Pittsburgh, PA, US

**Yuqing Tang**
Carnegie Mellon University, Pittsburgh, PA, US

**Point of Contact**
Paul R. Smart
Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
email: ps02v@ecs.soton.ac.uk

## Abstract

The aim of cognitive social simulation is to improve our understanding of the complex inter-play between factors that are spread across the cognitive, social and technological domains. This makes cognitive social simulation techniques particularly appealing as a means to undertake experiments into socially-distributed cognition. The current paper reports on the results of an ongoing effort to develop a cognitive social simulation capability that can be used to undertake studies into team cognition using the ACT-R cognitive architecture. The focus of the cognitive modeling effort associated with the development effort is a particular team-based problem solving task that forms part of the Experimental Laboratory for Investigating Collaboration, Information-sharing, and Trust (ELICIT) experimentation framework. This task has been used with human subjects to investigate the effect of different command and control organizational structures on collective problem solving performance. The results of the cognitive modeling effort are presented and future work to extend both the simulation capability and the cognitive model are outlined. By comparing the results obtained with the ACT-R simulation capability with those obtained from previous experiments using the ELICIT experimentation framework, it should be possible to evaluate the extent to which ACT-R agents exhibit performance profiles similar to those of their human counterparts. This will support the effort to evaluate the extent to which cognitive social simulation experiments with ACT-R can be used to generate findings of predictive relevance to future studies using the ELICIT experimentation framework.

# Towards a Cognitively Realistic Computational Model of Team Problem Solving Using ACT-R Agents and the ELICIT Experimentation Framework

Paul R. Smart[a], Darren P. Richardson[a], Katia Sycara[b] and Yuqing Tang[b]

[a]Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
[b]Robotics Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, US

## ABSTRACT

The aim of cognitive social simulation is to improve our understanding of the complex inter-play between factors that are spread across the cognitive, social and technological domains. This makes cognitive social simulation techniques particularly appealing as a means to undertake experiments into socially-distributed cognition. The current paper reports on the results of an ongoing effort to develop a cognitive social simulation capability that can be used to undertake studies into team cognition using the ACT-R cognitive architecture. The focus of the cognitive modeling effort associated with the development of the simulation capability is a particular team-based problem solving task that forms part of the Experimental Laboratory for Investigating Collaboration, Information-sharing, and Trust (ELICIT) experimentation framework. This task has been used with human subjects to investigate the effect of different command and control organizational structures on collective problem solving performance. The results of the cognitive modeling effort are presented and future work to extend both the simulation capability and the cognitive model are outlined. By comparing the results obtained with the ACT-R simulation capability with those obtained from previous experiments using the ELICIT experimentation framework, it should be possible to evaluate the extent to which ACT-R agents exhibit performance profiles similar to those of their human counterparts. This will support the effort to evaluate the extent to which cognitive social simulation experiments with ACT-R can be used to generate findings of predictive and explanatory relevance to future studies using the ELICIT experimentation framework.

**Keywords:** collective cognition, social information processing, distributed cognition, team cognition, social simulation, group performance, cognitive modeling, cognitive architecture, ACT-R

## 1. INTRODUCTION

In recent years, the socially-situated and socially-distributed nature of human cognition has become an increasing focus of research attention for the cognitive scientific community [1, 2, 3, 4, 5]. Cognitive processes that were typically studied at the level of individual agents, such as memory, are now being re-examined within a more social context [6], and increasing attention is being paid to the factors that enable groups to function as the processors of information [7]. Research within this area tends to go under a broad array of headings, such as group cognition [4], team cognition [8], distributed cognition [9], and collective cognition [3, 10]; however, what unites all these terms is an interest in the way in which social and cognitive factors work to influence performance outcomes at both the individual and collective levels. These issues are, of course, of considerable importance in military contexts, where many forms of problem solving and decision-making activity rely on the coordinated interaction of multiple individuals. Military command and control capabilities, for example, are often ones that are influenced by factors that extend across the cognitive, social and technological domains, and this underlies the need to understand the effect of these factors on team performance (e.g., [3, 11, 12, 13]).

In the attempt to gain a better understanding of the factors that affect performance in socially-distributed information processing contexts, researchers have resorted to a variety of mathematical modeling and computer simulation approaches [14]. Among these, the use of multi-agent systems to study the dynamics of collective behavior has emerged as a particularly important paradigm. For the most part, these multi-agent studies (commonly referred to as social simulation studies) have relied on agents that feature little or no internal cognitive processing capabilities. Most of the commonly available social simulation tools (e.g., Swarm [15] and RePast [16]) assume that individual agents have very rudimentary cognitive processing capabilities, and in some cases, aspects of agent cognition (e.g., beliefs or attitudes) are represented by simple numerical values that do not respect the constraints imposed by the human cognitive system.

Further author information: (Send correspondence to Paul R. Smart)
Paul R. Smart: E-mail: ps02v@ecs.soton.ac.uk

In order to improve the cognitive sophistication and fidelity of social simulation experiments, Sun [17] has advocated the use of what are called cognitive architectures. Cognitive architectures are computational frameworks that make particular commitments about the kinds of mental representations and computational procedures that are sufficient to explain important aspects of human cognition, such as problem solving, memory and learning [18]. As a result, cognitive architectures can be used to impose realistic constraints on the cognitive capabilities of synthetic agents in the context of social simulation experiments. The integration of cognitive architectures into social simulation results in what Sun [17] refers to as *cognitive social simulation*. By factoring cognitive architectures into social simulation experiments, Sun argues that we are provided with the opportunity to study the interaction between social and cognitive factors; for example, we can study the effect that different cognitive variables (such as memory decay rates, learning rates, attention, and so on) have on aspects of collective performance. Cognitive architectures thus enrich the range of experimental opportunities that are open to investigators, and they also support the attempt to better understand the interaction between cognitive, social and technological factors in episodes of collective problem solving.

The current paper describes an ongoing effort to develop a cognitive social simulation capability based around a particular cognitive architecture, called Adaptive Control of Thought-Rational (ACT-R) [19, 20]. Although ACT-R is one of the most widely used cognitive architectures, there are relatively few studies that use ACT-R to study cognition in social contexts (one of the few exceptions is a recent study by Reitter and Lebiere [21] that applied ACT-R to a social information foraging task). Given the aforementioned importance of socially-situated cognition as a focus area for scientific study, coupled with the need to incorporate cognitively realistic design constraints into multi-agent simulations, there is a compelling need to better understand how to apply cognitive architectures to team-based situations. The specific aim of our research is to develop a generic capability for running cognitive social simulation experiments and then apply this capability to a specific task. The capability we aim to develop is hereafter referred to as the ACT-R Cognitive Social Simulation Capability or ACT-R/CSSC, and the task that is the focus of our current modeling and simulation efforts is a task that has been previously used to investigate socially-distributed problem solving as part of what is called the ELICIT experimentation framework. ELICIT, in this case, is an acronym that stands for the Experimental Laboratory for Investigating Collaboration, Information Sharing and Trust. It represents a sustained effort to advance our understanding of the factors that affect collective performance in task contexts that incorporate both cognitive and social elements. The task that is commonly used as part of the ELICIT experimentation framework – the ELICIT task – is one that requires individuals to reason over bodies of task-relevant information and to also share information with others in the context of a distributed, network-enabled task environment. The ELICIT task is thus one that draws on factors that are spread across the cognitive, social and technological domains. These features make the task suitable for testing the ability of the ACT-R/CSSC to support experimental studies of team-based problem solving behavior.

This paper describes our progress with respect to the implementation of the ACT-R/CSSC, and it also describes our attempt to support cognitive social simulations with respect to the ELICIT task. Following a brief overview of the ACT-R cognitive architecture in Section 2, the ELICIT task (and associated experimentation framework) is described in Section 3. Section 4 presents the various components of the ACT-R/CSSC, and describes how these components can be used to support cognitive social simulation experiments with the ELICIT task. A cognitive model that enables ACT-R agents to reason over ELICIT-related information is described in Section 5. For convenience, we refer to this model as the ELICIT Computational Cognitive Model (ECCM). The basic operation of the ACT-R/CSSC is described in Section 6, and the focus of our future work efforts is outlined in Section 7. The work described in this paper is being undertaken as part of the International Technology Alliance (ITA) program, which is a consortium of academic, industrial and government partners undertaking fundamental research in the network and information sciences. A specific aim of the ITA program is to deliver research outcomes that support military coalition operations involving a multiplicity of military services from different nation states.

## 2. ACT-R

Sun [17] defines a cognitive architecture as "a domain-generic computational cognitive model that captures essential structures and processes of the individual mind for the purpose of a broad (multiple domain) analysis of cognition and behaviour" (p. 33). A cognitive architecture is thus a framework that captures some of the relatively invariant features of the human cognitive system – those features that are deemed to be more-or-less constant across domains, tasks and individuals. One example here is the mechanisms that support the storage and retrieval of information from long-term memory. Although a number of features of the task environment may affect the ability of subjects to recall information, the mechanisms that actually realize the recall process are unlikely to change from one task to another. In discussing cognitive architectures within the cognitive scientific literature, authors often draw an analogy between a cognitive architecture and the architecture for a building or computer system [17, 19, 22]. The architectural specification of a building tends to focus on some details at the expense of others. It includes, for example, permanent and enduring features, such as the building's foundation, the location of windows, and the dimensions of rooms. It does not, however, include features that are easily rearranged or replaced,

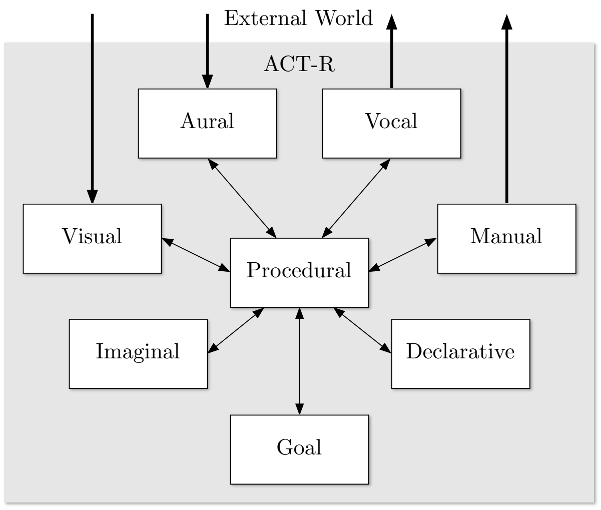


Figure 1: The core modules of the ACT-R v.6 cognitive architecture.

such as furniture and appliances. Similarly, a cognitive architecture tends to focus on particular features that constitute the core elements of cognition. These include the mechanisms to represent, reason over, and remember information.

In applying a cognitive architecture to a particular aspect of cognitive behavior, a modeler endeavours to develop a cognitive model that includes task- and domain-variant elements. This includes the information that is to be processed as part of the task, the rules that govern behavior and the background knowledge that needs to be used in performing the task. This model draws on the generic representational schemes and computational procedures of the cognitive architecture in order to support its execution. The result is that a number of task-specific cognitive models can be developed that all draw on the same underlying (cognitive) mechanisms. This is helpful in terms of standardizing research into cognitive modeling, and it also affords a degree of explanatory unification in that a number of otherwise disparate behaviors can be accounted for in terms of a common set of underlying mechanisms.

ACT-R is one of a number of cognitive architectures that have been used for cognitive modeling [19,20]. It is primarily a symbolic cognitive architecture in that it features the use of symbolic representations and explicit production rules; however, it also makes use of a number of subsymbolic processes that contribute to aspects of performance (see [20]). The ACT-R architecture has been used to model human cognitive performance in a wide variety of experimental contexts[*], and it has generated findings of predictive and explanatory relevance to hundreds of phenomena encountered in the cognitive psychology and human factors literature. This has earned it a reputation as the cognitive architecture that is probably the "best grounded in the experimental research literature" [24, p. 24]. ACT-R has also been used to model performance in a range of complex task settings. For example, ACT-R has been used to model driver behavior [25], including the effects of concurrent activities (such as cell phone usage) [26] and sleep deprivation on driver performance [27]. These features (widespread use, predictive and explanatory power, applicability to complex tasks) make ACT-R a compelling target for running cognitive social simulation experiments. To date, however, few studies have sought to apply ACT-R to situations involving socially-distributed information processing.

From an architectural perspective, ACT-R consists of a number of modules (see Figure 1), each of which is devoted to processing a particular kind of information. Each module is associated with a capacity-constrained buffer that can contain a single item of information, called a chunk. The modules are assumed to access and deposit information in the buffers, and coordination between the modules is achieved by a centralized production system module – the procedural module – that can respond to the contents of the buffers and change buffer contents (via the execution of production rules). Importantly, the procedural module can only respond to the contents of the buffers; it cannot participate in the internal encapsulated activity of modules, although it can influence such processes. The inspiration for ACT-R's modular design is based on a combination of functional constraints and recent advances in neuroscience concerning the localization of specific brain functions. Some of the modules within ACT-R are deemed to correspond to particular brain regions, and this has enabled ACT-R modelers to make predictions concerning the temporal profile of brain activity during aspects of task performance (see [28]).

As shown in Figure 1, there are eight core modules in the latest version of ACT-R:

[*]The ACT-R website [23] provides access to a broad range of academic publications covering areas such as problem solving, learning, language processing, decision making, and perceptual processing.

- **Input and Output Modules.** There are four input/output modules (Visual, Aural, Vocal and Manual). These provide support for modeling agent-world interactions.
- **Goal Module.** Actions within ACT-R are often dependant upon the current goal being pursued. The goal module is a specialized form of memory, with its own buffer, and it stores the current state of these goals.
- **Imaginal Module.** The imaginal module is responsible for manipulating intermediate representations of a problem when working towards a goal. For example, when calculating $x$ in $x - 4 = 7$, the intermediate stage $x = 7 + 4$ can be stored in the imaginal module before being evaluated.
- **Declarative Module.** This module implements the memory system of the agent. It stores information in the form of chunks, each of which are associated with activation levels.
- **Procedural Module.** The procedural module is responsible for coordinating between the other modules. It contains rules that fire in response to the contents of the module buffers. As mentioned above, the contents of the various modules are typically changed as a result of rule execution.

These modules (and their associated buffers) tend to form the basis of most ACT-R models. Cognitive modelers are not, however, restricted to the use of these modules, and new modules can be added to implement additional functionality. As an example of this kind of extension of the default ACT-R architecture, Rodgers et al [29] added a total of nine buffers to the ACT-R architecture as part of their effort to implement a situation model (corresponding to a "mental model of the objects, events, actions, and relationships encountered in a complex task simulation" [29, p. 313]). The simulation capability developed to support experimental simulations in the context of the ELICIT task (see Section 4) also relies on a number of custom modules. The functionality of these modules is described in Section 4.2.

It is important to note that ACT-R is not the only cognitive architecture available; other architectures include State, Operator and Result (SOAR) [30, 31, 32] and Connectionist Learning with Adaptive Rule Induction ON-line (CLARION) [17, 33]. Given the range of cognitive architectures available, it is important to consider the factors that motivate the use of one particular cognitive architecture, namely ACT-R, to study issues associated with task performance in team-based contexts. One of the main reasons for selecting ACT-R, in this respect, relates to its widespread use in psychological modeling. As mentioned above, ACT-R has a long history of use in terms of the computational modeling of psycho-cognitive processes. In addition to this, ACT-R has been used in a number of militarily-relevant application areas (this is relevant given the focus of the ITA program on military coalition operations). Best and Lebiere [34], for example, discuss the use of ACT-R to implement synthetic agents that engage in urban combat in virtual environments as part of Military Operations on Urban Terrain (MOUT) training simulations. ACT-R is also being used to develop language-enabled synthetic teammates to support training exercises in three-person Unmanned Aerial Vehicle (UAV) control teams [35]. Finally, ACT-R has been used to study the interactions between cognitive and social factors in a group problem solving task [21]. Using an information foraging task that required multiple ACT-R agents to search for the location of specific information items, Reitter and Lebiere [21] were able to study the effect of artificial manipulations of memory (specifically, the rate at which the activation of information items in memory decayed) on aspects of collective performance. This study helps to demonstrate the applicability of ACT-R to cognitive social simulation experiments: it demonstrates that ACT-R can be applied to situations in which both social and cognitive factors are relevant to aspects of task performance.

## 3. ELICIT EXPERIMENTATION FRAMEWORK

ELICIT denotes an ongoing effort to provide a common experimental framework to investigate issues in group-level problem solving [36]. The ELICIT experimental framework consists of the ELICIT experimental platform, which corresponds to the software environment responsible for running ELICIT experiments; the ELICIT experimental task, which is the actual task performed by experimental participants; and the ELICIT factoids, which are the actual pieces of task-relevant information that are processed by experimental participants during the course of task execution. The ELICIT experimental platform consists of the various computer interfaces presented to human subjects during the ELICIT task. These interfaces comprise what we refer to as the task environment, and their importance (from the perspective of cognitive modeling) is discussed in Section 7.

The ELICIT task has been the main focus of our recent modeling efforts in the ITA program (see Section 5). The task involves the selective presentation of information items – called factoids – to experimental subjects. Each factoid provides a limited amount of information about a situation, and the aim of the subject is to assimilate enough information in order to make a decision regarding the features of an impending terrorist attack (these features are typically referred to as the 'dimensions' of the ELICIT task). The particular features the subject needs to resolve are as follows:

- **who:** the group that will attempt to perform the attack

- **where:** the country in which the attack will take place
- **what:** the kind of target the attack will be against (e.g., an army base)
- **when:** the date and time of the attack (specifically, the day, month and hour of the attack)

In a typical ELICIT experiment, a team of human subjects ($N = 17$) are presented with a computer interface that provides access to a subset of the total set of factoids used for the experiment (the total set of factoids used for an ELICIT experiment is typically referred to as a 'factoidset'). A limited number of factoids are distributed to subjects at specific intervals in the experiment, and these are referred to as 'distribution waves'. Assuming that the experiment consists of 17 subjects, two factoids are distributed to each subject in the first distribution wave, with a further two factoids distributed to each subject in two successive distribution waves. The result is that by the third distribution wave, all factoids ($N = 68$) in the relevant factoidset will have been made available to the human team. The factoids are delivered to each team member's 'inbox', which acts as a repository of all the factoids that an individual team member can access via the aforementioned computer interface. In order to solve the ELICIT problem, subjects need to share factoids with one another. This is because the four factoids that are delivered to each subject via the distribution waves do not contain sufficient information to solve the problem (i.e., generate answers to the four dimensions of the problem). Subjects therefore need to share the factoids they have access to (i.e., the factoids in their inbox) by either sending factoids to other experimental participants (via a 'send action') or posting factoids to one of a number of 'websites' (via a 'post action'). The result of a send action, in this case, is that the factoid sent by the originating agent gets placed in the inbox of the receiving agent. In the case of a post action, the factoid that gets placed in a shared repository that can be accessed by any team member that has sufficient access rights to the repository (an ELICIT website thus serves as a mechanism for making factoids available to a larger community of team members than is possible with a single send action). Team members gain access to the factoids associated with websites via what are called 'pull actions'. These actions retrieve the factoids from the website repository and place the factoids in the team member's inbox. The profile of information sharing in the task is a function of both agent characteristics (e.g., the agent's willingness to share information), as well as the structure of the organizational environment in which the task is performed. For example, a team member's access to particular websites can be controlled through the setting of 'access rights'. In addition, the structure of the communication network can be configured to limit the channels of communication between team members. In the case of a fully-connected communication network, each team member can send messages to every other member of the team directly; however, more restrictive communication policies can be created to limit the profile of inter-agent communication. In general, the accessability of the websites (website access policy) and the structure of the communication network (communication policy) are configured so as to investigate the effect of different organizational environments on the team's ability to solve the ELICIT problem (e.g., [11]).

In addition to the studies with human subjects, a synthetic agent has also been developed to support multi-agent simulations of the ELICIT task [37]. Such agent-based simulations can be run with an extension of the ELICIT experimental framework, which is referred to as the agent-based ELICIT (abELICIT) framework. The synthetic agents used in the context of such simulations are typically referred to as abELICIT agents. Importantly, the use of the abELICIT experimentation framework can be used to extend the range of experiments of that can be performed with the ELICIT experimentation platform [37]. In addition, since it is possible to tailor the behavior of the synthetic agents to resemble that of human subjects [38], it is possible to run experiments that involve a mixture of both human and synthetic agents.

Although abELICIT agents are clearly intended to function as virtual surrogates of human agents, they are not based on a cognitive architecture that is modeled after the human cognitive system. This makes it difficult to run simulation experiments in which the role of specific cognitive variables is systematically investigated (for example, it is difficult to run experiments that explore the effect of different cognitive strategies or different mnemonic capabilities). A key aim of our research in the ITA program is to support the implementation of ELICIT experiments in such a way that the effect of different cognitive variables can be examined. In addition, we wish to explore the potential interactions that exist between agent-level cognition and the features of the informational, social and technological environment in which ELICIT-related problem solving occurs (see [39]).

One of the main advantages of focusing on the ELICIT task for the purposes of cognitive modeling and cognitive social simulation is that it provides a previously well-studied task that is associated with standard bodies of task-relevant information (i.e., factoidsets). The availability of the factoidsets reduces the effort required to engineer a cognitive model in the sense that it both reduces the overhead associated with the creation of task-relevant materials (which are used for testing purposes), and it also provides insight into the kind of knowledge structures that an agent needs in order to solve the ELICIT problem. The first 10 factoids from one particular ELICIT factoidset (namely 'factoidset4aGMU') are shown in Table 1 (there are 68 factoids in the full factoidset). As can be seen from this subset of factoids, each factoid provides information about the entities and relationships associated with a particular situation, and at least some of these factoids support inferences that enable particular kinds of suspect entities (e.g., groups, countries and targets) to be eliminated. An example is provided by factoids 1 and 2 from Table 1. Factoid 2 states that the impending attack will be a suicide bombing attack, and we learn from factoid 1 that Gray and Teal groups do not employ suicide bombing tactics. As a result of being presented with these two pieces of information we can infer that neither the Gray nor the Teal group can be involved in the attack, and they can thus be eliminated from the list of suspect groups. We can also see from Table 1 that factoids come in four basic types: essential, key, supportive and noise (see [36]) (these are indicated by the letters 'E', 'K', 'S', and 'N' in Table 1). Expert and key factoids provide important information that is relevant to the process of resolving the who, what, when and where aspects of the task; supportive factoids provide information that tends to support the information contained in the key and essential factoids; and noise factoids contribute nothing to an agent's ability to solve the problem – the problem can be solved even if these factoids are ignored.

Table 1: Subset of factoids from one of the factoidsets (namely 'factoidset4aGMU') used in ELICIT experiments. The characters in the 'Type' column specify the type of the factoid: E = Essential, K = Key, S = Supportive and N = Noise.

| # | Type | Factoid |
|---|---|---|
| 1 | E | The Gray and Teal groups do not employ suicide bombers |
| 2 | E | There will be a suicide bomber attack at a school |
| 3 | E | The Silver group does not work in Pi |
| 4 | E | The Silver group only attacks during the day |
| 5 | N | The Rose group may be involved |
| 6 | K | The Sienna and Rose groups only target the military |
| 7 | S | Reports from the Teal group indicate standard levels of activity |
| 8 | N | There is a lot of activity involving the Rose group |
| 9 | N | The Gray group is recruiting locals – intentions unknown |
| 10 | K | The Turquoise group focuses on destroying energy infrastructure |

Aside from the availability of factoidsets, which reduce the overhead of creating task-relevant data and support the process of model development, a number of other factors serve to make the ELICIT task a compelling target for cognitive social simulation. These include the fact that the ELICIT task has been used in a large number of studies undertaken with both human and synthetic agents. Such studies yield bodies of empirical data that can be used to evaluate the results of ACT-R-based cognitive social simulation experiments. The studies with human subjects are of particular importance in this respect. They provide some indication as to how the performance of human teams varies with specific experimental conditions, and they thus provide a metric against which ACT-R-based studies can be evaluated. Inasmuch as the ACT-R cognitive models provide a faithful representation of the cognitive processes of human subjects in particular task contexts, we should expect to see a similar performance profile in the case of both ACT-R simulations and human team experiments[†].

Unfortunately, the decision to adopt ELICIT as the target for cognitive modeling efforts does not come without a host a worries and concerns. A particular worry relates to the complexity of the ELICIT task itself and the extent to which ACT-R, which is typically used to model what might be called micro-cognitive competencies (e.g., the cognitive processes associated with psychophysical judgments or list memory tasks), can scale up to a task that involves sophisticated forms of agent-world interaction, the manipulation of complex bodies of domain-relevant knowledge, the implementation of complex inferences, and the processing of linguistic content (in the form of factoids). All of this, of course, occurs in the context of a team-based problem solving activity that features a range of complex decisions about when to share information and who to share information with. Such complexity presents a formidable challenge to any ACT-R modeling effort. Our approach to this challenge is to develop the cognitive model for the ELICIT task in a series of stages, first focusing on the ability to reason over situation-relevant information and then focusing on the cognitive processes that underlie decisions about information sharing. In addition, it is important to note that ACT-R has been used to model performance in a range of complex task settings [40, 41], including ones that feature the sharing of information with multiple team members [21]. As a result, there are a range of existing models that can be used as a point of departure in our own computational cognitive modeling efforts.

Another problem that is associated with the ELICIT framework concerns the nature of the task itself and the extent to which the features of the task resemble those seen in real-world problem solving contexts. In particular, the ELICIT task is intended to capture at least some of the features associated with intelligence analysis processes. However, as has been noted in previous work (e.g., [42]), there are a number of important differences between the nature of the problem encountered in the context of the ELICIT task and the problems dealt with by intelligence analysts. Perhaps most importantly, the fact that a fixed set of factoids are used in the ELICIT task, coupled with the fact that these factoids are complete (in the sense that they enable a subject to derive the correct answer to the problem) and are always assumed to be true, constitutes a significant

[†]Much, of course, depends on the extent to which the features of the task environment and experimental condition are also adequately represented within the context of the experimental simulations (see Section 7).

departure from the reality of most intelligence analysis situations: in most real-world intelligence analysis problems agents have to deal with conflicting, uncertain and dynamic information from a constantly evolving situation. Future work should probably aim to consider modifications to the ELICIT factoidsets in order to establish a closer alignment to the kind of features seen in more naturalistic contexts. A particular point of interest in regard to our own work concerns the need to explore dynamic situations, ones in which the veracity of particular factoids can be negated by the receipt of subsequent information. One reason why this is important is that it may serve to reveal the *positive* functional role played by an apparent shortcoming of the human cognitive system, namely, our tendency to forget information. In dynamic information environments, a capacity to forget information may result in more recent information having greater saliency at the expense of more outdated information. In particular, once agents are endowed with an ability to forget information, the role that outdated information plays in maintaining collective interpretations of the environment may be undermined (see [39]). This possibility serves to highlight the importance of simulation experiments using cognitive architectures, such as ACT-R. By factoring in the kinds of constraints and capabilities exhibited by the human cognitive system, we are able to assess the role that cognitive-level variables play in supporting (or subverting) task performance. Conventional agent-based simulations often fail to yield this sort of understanding because they fail to adequately represent the processing characteristics of the human cognitive system, and this is one reason why we have elected to use ACT-R agents rather than abELICIT agents in the context of our own simulation experiments with the ELICIT task: ACT-R provides us with a means to gain a better understanding of the potential interactions between the human cognitive system and the social, technological and informational context in which socially-distributed cognition occurs (see [39]).

A final limitation of the ELICIT task concerns the potential ambiguity of the factoids used in the task. In our effort to develop a cognitive model to support the ELICIT task (see Section 5), we noted some degree of ambiguity in at least some of the ELICIT factoids. One example of this is described by Smart and Sycara [43] in the context of one particular factoidset, namely 'factoidset4aGMU'. The particular problem relates to factoid 51, which reads 'Sigma has closed all its schools'. This factoid is intended to rule out Sigma (a country) as a suspect entity on the grounds that we know (from other factoids) that the attack will be a school, and in order for a country to be a suspect it must (logically) contain schools. The problem with factoid 51, in this case, is that the closure of schools is intended to mean that there are no schools in Sigma that are *viable* targets and Sigma should therefore be eliminated from the list of suspect entities. As is highlighted by Smart and Sycara [43], this particular interpretation is not always guaranteed to occur, and this perhaps serves to highlight one of the shortcomings of the current ELICIT factoidsets: they require subjects to make particular kinds of interpretations, but not all of these interpretations are enforced by the semantics of the statements themselves. Obviously, further work is required to investigate this particular issue. Ideally, one would like the ACT-R agents to interpret factoids in the same way as human subjects, and thus one strategy could be to modify the text of the factoids in order to reduce semantic ambiguity. A second strategy could involve an effort to record the kinds of interpretational errors made by human subjects when they encounter the factoids and then ensure that ACT-R agents make the same sort of errors with similar frequency. In both cases, one can imagine collecting the required data with questionnaires that present each of the factoids and then solicit input from the respondent in the form of (e.g.) multiple choices.

## 4. ACT-R COGNITIVE SOCIAL SIMULATION CAPABILITY

In order to use ACT-R as a platform for cognitive social simulation, a number of extensions were made to the core ACT-R architecture. The most important extension, from the perspective of cognitive social simulation, relates to the creation of a specialized 'messaging' module that enables agents to exchange text messages with other agents. A number of other custom modules were created in order to extend the ACT-R architecture in order to support the execution of simulation experiments similar to those that have been undertaken with the ELICIT experimentation framework. For example, the 'web' module (see Section 4.2.3) is intended to emulate the functionality of the ELICIT experimentation platform in respect of the ability to pull and post messages to various websites. All the modules associated with the ACT-R/CSSC are described in Section 4.2.

In addition to the custom modules, a number of memory-resident Lisp 'databases' were created to support the storage of information relating to both the task and simulation experiment. These databases are implemented within the same Lisp environment as that hosting ACT-R. The result of this implementation strategy is that relevant information in the databases can be accessed directly from within ACT-R without relying on external, third-party components. The databases forming part of the ACT-R/CSSC are described in Section 4.1.

Together, the custom ACT-R modules and databases provide the necessary infrastructure to run a range of cognitive social simulation experiments using ACT-R. ACT-R cognitive models that implement task-specific cognitive processing routines (e.g., the ECCM – see Section 5) can make use of this infrastructure in order to support inter-agent communication and control the access of agents to specific network-accessible information repositories (i.e., ELICIT websites).

## 4.1 Databases

The ACT-R/CSSC relies on a variety of databases to store information relating to agent characteristics, agent behaviour, communicative exchanges and the structure of the agent communication network. The function of each of these databases is described in Table 2, and their structure is presented in Table 4 (see Appendix A). A common set of Lisp macros and functions is available to support the process of querying, creating, updating and deleting records from each of the databases.

Table 2: Databases associated with the ACT-R/CSSC.

| **Database** | **Description** |
|---|---|
| Message Database | Contains all the messages that are posted to agents either from other agents or from the ACT-R/CSSC. The name of the intended recipient is stored with each message record, and this allows the message database to function as a repository of 'inboxes' for all of the agents in the simulation. |
| Web Message Database | Contains messages that are posted to particular ELICIT websites, such as the 'who-website' or the 'where-website'. |
| Connection Database | This is used to define the structure of the communication network that exists between agents. This database contains records that specify the direct channels of communication that exist between agents. |
| Triple Database | This is used to store general information about the experimental simulation. For example, the triple database can be used to store information about the number of agents to create, the properties of agents, the particular factoidset to use, the time allowed for agents to complete the target task, and so on. The triple database stores information in the form of subject, predicate, and object triples, similar to those encountered in the context of the Resource Description Framework (RDF) [44]. |
| Factoid Database | This database is preloaded with all the factoids contained in a particular factoidset. It is used to distribute particular factoids to particular agents at specific times within the simulation. This corresponds to the way in which factoids are distributed to experimental participants in a series of 'distribution waves' during the course of ELICIT experiments. |
| Action Database | Contains information about the various actions that are made by agents during the course of the simulation. This includes the actions of posting messages to websites (post actions), pulling messages from websites (pull actions) and sending messages (send actions) to other agents. |
| Identification Database | Contains information about the identification actions made by agents. An identification action is recorded whenever an agent attempts to identify the who, what, where and when dimensions of the ELICIT problem (see Section 3). |

At the conclusion of a particular simulation experiment, the ACT-R/CSSC automatically serializes the contents of all databases to text files and saves these files in a directory on the host machine. These files can then be processed by other applications to support the analysis and visualization of simulation results (see Section 6).

## 4.2 Modules

As mentioned in Section 2, ACT-R modules provide one way in which the functionality of the core ACT-R architecture can be extended. In the case of the ACT-R/CSSC, a number of custom modules were developed and integrated into the ACT-R architecture. These include the 'messaging', 'language', 'web' and 'self' modules. The function of these modules is to support the execution of cognitive social simulation experiments in which agents must exchange information (in the form of simple text messages) as part of some collaborative episode of problem solving activity in a network-enabled environment. These modules provide the core functionality that is required to run experiments similar to those undertaken with the ELICIT experimentation platform.

### 4.2.1 Messaging Module

The messaging module features two buffers: 'send-message' and 'receive-message'. The send-message buffer enables agents to post a message to other agents in the simulation, while the receive-message buffer enables agents to check for unread messages and retrieve any messages that are available. A particular database – the 'message database' – is used to store all the messages sent by agents, and the messaging module interfaces with this database in order to both create new messages and retrieve existing ones. Together, the messaging module and the message database implement the capability for agents to communicate with one another as part of a socially-distributed problem solving process.

The messages that get communicated by agents are represented by particular kinds of chunks, called `message` chunks. These chunks contain the following slots:

- **source:** Specifies the name of the agent that posted the message.
- **target:** Specifies the intended target of the message. If the value of this slot is 'any', then the message will be posted to any agent that the originating agent can communicate with based on the structure of the communication network (as specified in the connection database).
- **text:** Specifies the text of the message to be communicated. In the case of ELICIT-based experiments, this slot contains the text of a particular factoid.
- **factoid-number:** Specifies the number of the factoid that is represented by the message chunk.

Whenever a `message` chunk is asserted in the send-message buffer as a result of rule execution, the messaging module first determines whether the target agent is a peer of the originating agent (this information is stored in the connection database). If this is the case, the messaging module will create a new record in the message database that reflects the information content of the `message` chunk. All such records are initially marked as 'unread', reflecting the fact that they have not been processed by the intended recipient.

Whenever an agent wants to retrieve new messages from the database, they can make a request to the receive-message buffer. This will cause the messaging module to check the message database for any unread messages that have been sent to the agent. If any such messages are available, one of the messages will be retrieved and a new `message` chunk will be created in the receive-message buffer of the relevant agent. The manner in which unread messages are selected from the database can be controlled by a particular parameter, called the 'message-selection-mode' parameter, which is associated with the messaging module (for example, the most recently posted messages can be selected by setting the message-selection-mode parameter to 'newest'.). Listing 1 presents two production rules that exemplify the process of sending and receiving messages via the buffers of the messaging module.

Listing 1: Two production rules that exemplify the use of the messaging module to send and receive messages via the send-message and receive-message buffers. The request to the parse-message buffer in the `interpret-and-send-received-message` rule causes the language module (see Section 4.2.2) to interpret the received message and create corresponding `statement` chunks in the agent's declarative memory module (see main text for details).

```
(P retrieve-unread-messages
    =goal>
        isa       sensemaking-goal
        task      process-messages
    ?receive-message>
        unread    true
        buffer    empty
    ==>
    +receive-message>
        isa       message
)

(p interpret-and-send-received-message
    =goal>
        isa       sensemaking-goal
        task      process-messages
    =receive-message>
        isa       message
        text      =text
    ?send-message>
        state     free
        buffer    empty
    ==>
    +parse-message> =receive-message
    +send-message>
        isa       message
        text      =text
        target    any
)
```

### 4.2.2 Language Module

In the case of ELICIT-based simulations, the `message` chunks that are processed by the messaging module serve as a vehicle for the communication of particular factoids. The `text` slot of each `message` chunk will therefore contain the text of a single factoid, and the `factoid-number` slot will contain the number of the factoid in the relevant factoidset. In order for agents to exploit the factoids and solve the ELICIT problem, the text of the messages must be converted into chunks that correspond to the internal mental representations that are used by ACT-R agents as part of some reasoning process. In the case of the ECCM presented in Section 5, ACT-R agents encode the information content of the factoids in the form of `statement` chunks. These are intended to represent the basic facts that are implied by a factoid, although they can also be used to represent the facts that are inferred by agents as a result of processing a factoid (see Section 5 for a more detailed discussion of `statement` chunks).

The process of converting `message` chunks into `statement` chunks is accomplished using the 'language' module of the ACT-R/CSSC. Like the messaging module, this module is a new custom module that does not form part of the core ACT-R architecture (see Figure 1). The language module exposes a single buffer, called 'parse-message' that can receive `message` chunks, 'interpret' the text content of the chunks (i.e., the text contained in the `text` slot of the `message` chunk), and create `statement` chunks reflecting the information content of the message. At the present time, the interpretation process is a simple one: the language module invokes a Lisp function, called `process-factoid`, which accepts the text of the factoid and the factoid number as input and then conditionally creates `statement` chunks based on the value of either of these parameters. The `process-factoid` function is specific to whatever factoidset is used in the simulation, so different versions of the function need to be created to cater for different factoidsets. A subset of the code associated with one version of the `process-factoid` function is shown in Listing 2.

Listing 2: Part of the code associated with the `process-factoid` Lisp function. This function converts factoid messages into `statement` chunks that represent an agent's understanding of the current situation.

```
(defun process-factoid (text-val factoid-number)
   "Converts ELICIT factoids into corresponding statement chunks."
   (cond
      ;; "The Gray and Teal groups do not employ suicide bombers" (factoid 1)
          ((string-equal (write-to-string factoid-number) "1") (add-dm-fct (list
                `(isa statement object gray-group
            attribute is-suicide-bomber value no source ,source-val)
                `(isa statement object teal-group
            attribute is-suicide-bomber value no source ,source-val)
          )))
          ;; "There will be a suicide bomber attack at a school" (factoid 2)
          ((string-equal (write-to-string factoid-number) "2") (add-dm-fct (list
                `(isa statement object attack
            attribute is-suicide-bombing value yes source ,source-val)
                `(isa statement object school
            attribute is-involved value yes source ,source-val)
                `(isa statement object attack
            attribute is-attack-against-school value yes source ,source-val)
          )))
      ;; Code to process remaining factoids goes here.
   )
)
```

As can be seen from Listing 2, the `process-factoid` function relies on the use of the ACT-R `add-dm-fct` function to create `statement` chunks and make these available to ACT-R agents. The `add-dm-fct` function is used to assert `statement` chunks directly into the agent's declarative memory module. This means that the chunks exist within the agent's memory, and they can be accessed by making retrieval requests to the declarative module as part of a cognitive processing routine, e.g., as part of the reasoning process used by agents to solve the ELICIT problem (see Section 5). The use of the `add-dm-fct` function, in this context, allows for situations in which multiple `statement` chunks must be asserted to reflect the factual content of a message (as can be seen from Listing 2, there is not always a one-to-one mapping between the messages that represent a factoid and the `statement` chunks that represent the semantic content of the factoid).

### 4.2.3 Web Module

The ELICIT experimentation framework makes use of 'websites' as part of the ELICIT task. These websites act as repositories of factoids that can be shared with any agent that has access to the website as part of the specific experimental protocol that is used. In the context of an ELICIT experiment, an agent can share information with other agents by posting factoids to a particular website, and they can retrieve factoids posted to the website by making a retrieval request against the website. In terms of the ACT-R/CSSC, agents interact with websites using the 'website' module. The functionality of this module is similar to that exhibited by the messaging module. In particular, the module exposes two buffers – post-message and pull-message – each of which enables agents to either post a message to a particular website or retrieve ('pull') a message from a particular website. In order to post a message, ACT-R agents first make a request to the post-message buffer to create a `message` chunk corresponding to a particular factoid[‡]. This `message` chunk is then stored in the web message database (see Section 2). When agents want to retrieve or 'pull' messages from a particular website, they make a request to the pull-message buffer using a `web-request` chunk. This chunk contains a slot, called `address` that specifies the particular website the agent wants to retrieve messages from. The result of making this request is that any messages associated with the relevant website are retrieved and asserted into the message database for the relevant agent. This means that following a retrieval request, the messages associated with a website are made available to an agent via the messaging module; i.e., agents can retrieve the new messages by making requests to the receive-message buffer of the messaging module (see Section 4.2.1). Listing 3 presents a sample production rule that demonstrates the process of posting and pulling messages to particular websites.

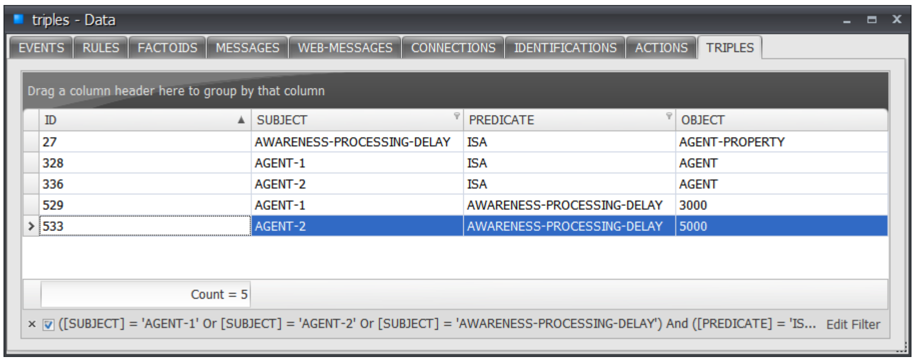
Listing 3: ACT-R production rule to demonstrate website interaction. The rule illustrates how agents can retrieve (or 'pull') messages from a particular website (in this case, the 'where-website') by making a request to the pull-message buffer. The rule also shows how agents can post a message to a particular website (in this case, the 'who-website') using the post-message buffer. Note how the `target` slot of the `message` chunk is used to post the message to a particular website.

```
(P pulling-and-posting-messages
    =goal>
        isa          sensemaking-goal
        task         process-messages
   =receive-message>
        isa          message
        text         =text
        factoid-number =number
   ==>
   +pull-message>
        isa          web-request
        address      where-website
   +post-message>
        isa          message
        text         =text
        factoid-number =number
        target       who-website
)
```

### 4.2.4 Self Module

The main aim of the abELICIT experimentation framework is to yield multi-agent social simulations that emulate some aspects of human performance in the context of the ELICIT task. This enables abELICIT agents to function as virtual surrogates of their human counterparts in the context of simulation experiments run solely with abELICIT agents or with a combination of abELICIT agents and human subjects. The behavior of abELICIT agents is controlled by a number of parameters that are incorporated into the abELICIT experimentation framework (at the time of writing, 54 such parameters are available). These parameters can be used to modify the processing and response characteristics of abELICIT agents in order to approximate the performance profile of particular human individuals [38]. In the context of the ELICIT experiments described by Manso [11, 42], for example, the abELICIT parameters were used to define three categories of abELICIT agent, namely, high-performing, average-performing, and low-performing agents. These three performance profiles were obtained by adjusting the value of specific parameters in order to obtain the desired behavior. For example, relative to the average-performing agents, the parameter settings for high-performing agents improved the speed with

[‡]Agents select which website to post to by using the `target` slot of the `message` chunk. For example, in Listing 3 the `target` slot is set to the value 'who-website'. This means that the message will be posted to the website containing information relevant to the 'who' dimension of the ELICIT problem.



| ID | SUBJECT | PREDICATE | OBJECT |
|---|---|---|---|
| 27 | AWARENESS-PROCESSING-DELAY | ISA | AGENT-PROPERTY |
| 328 | AGENT-1 | ISA | AGENT |
| 336 | AGENT-2 | ISA | AGENT |
| 529 | AGENT-1 | AWARENESS-PROCESSING-DELAY | 3000 |
| 533 | AGENT-2 | AWARENESS-PROCESSING-DELAY | 5000 |

Count = 5

([SUBJECT] = 'AGENT-1' Or [SUBJECT] = 'AGENT-2' Or [SUBJECT] = 'AWARENESS-PROCESSING-DELAY') And ([PREDICATE] = 'IS...

Figure 2: A filtered view of the triple database showing the settings of the 'awareness-processing-delay' agent property for two agents, named AGENT-1 and AGENT-2.

which they could process information and enhanced their propensity to access information from websites (see [42] for more details). The abELICIT parameters are thus an important area of interest for those concerned with the development of cognitive social simulations that target the ELICIT task. Given that the parameters are used to obtain approximations to human performance in the context of abELICIT experiments, they provide important information about the kind of response characteristics that synthetic agents will need to possess. In addition, since one of our immediate aims in the ITA program is to compare the performance of ACT-R agents with both human subjects and abELICIT agents, as seen in the context of the experiments undertaken by Manso and others [11, 42] (see Section 7), it is important to be able to adapt the behavior of ACT-R agents with respect to the same sort of agent-level parameters as is used in those experiments.

The approach to incorporating abELICIT parameters into the ACT-R/CSSC involves the use of the 'self' module. The purpose of this module is to make information about the self, such as agent personality characteristics, available to the procedural module in order to influence the dynamics of rule execution. This is accomplished by programmatically creating a chunk in the buffer of the self module as part of the ACT-R agent instantiation procedure. In the case of simulations performed with the ACT-R/CSSC, this chunk contains slots that correspond to all of the parameters included in the abELICIT experimentation framework.

The chunk that is created in the buffer of the self module is an instance of the `self` chunk-type[§]. This is a chunk-type that is automatically created whenever an ACT-R model is instantiated (or reset). Unlike most ACT-R chunk-type specifications, the slots associated with the `self` chunk-type are not specified in advance of the simulation being run (i.e., they are not hard-coded into the ACT-R/CSSC environment or the ECCM). Because the ACT-R/CSSC is intended as a generic platform for cognitive social simulation, it is important that the slot structure of the `self` chunk-type is highly configurable. In addition, given the number of abELICIT parameters available ($N = 54$), the task of manually coding the `self` chunk-type specification and then configuring individual instances of the chunk-type for each agent is highly labor-intensive, especially for simulations that feature lots of agents with different parameter configuration profiles. In order to simplify the process of asserting and configuring agent parameters, the ACT-R/CSSC provides functions that enable an experimenter to define properties that should be included as slots in the `self` chunk-type specification at runtime. The ACT-R/CSSC triple database (see Section 4.1) is used for this purpose. In particular, the self module exposes a function that enables new agent properties to be defined as instances of what is called the AGENT-PROPERTY property type. Additional functions are then used to initialize and update the value of these properties for specific agents. The result is that all the necessary information required to create the `self` chunk-type and initialize slot values for individual ACT-R agents is available in the triple database prior to the initiation of the simulation experiment. Figure 2 provides a filtered view of the contents of the triple database for one particular agent property (in this case, the property corresponds to the 'awareness-processing-delay' abELICIT parameter) as well as the property setting for two agents, named AGENT-1 and AGENT-2. Whenever an ACT-R agent model is initialized (or reset), the slot structure of the `self` chunk-type specification is created based on all the AGENT-PROPERTY instances that are asserted in the triple database. The values associated with the AGENT-PROPERTY instances are then used to set the values of the slots of the individual `self` chunks (i.e., instances of the `self` chunk-type) that are created in the buffer associated with the self module.

During the course of the simulation, the conditional elements of the production rules contained within the ECCM can match to the contents of the self module buffer in order to adapt agent behavior in particular ways. For example, the rule in Listing 4 demonstrates how a rule can be used to control the timing of specific actions.

[§]A chunk-type, in this case, specifies the structure of a chunk in terms of the slots that it can contain (a chunk-type essentially provides a template for a chunk, in the same way that a class in object-oriented programming provides a template for the data structure of an object).

Listing 4: ACT-R rule demonstrating the use of the self module to control agent response characteristics. In this case, the value of the `cycle` slot of the chunk in the goal buffer is bound to a variable named `=time`. This variable records the time that has elapsed since the simulation was first started. When the value of the `=time` variable matches the value contained in the `ready-interval-delay` slot of the chunk contained in the self buffer, the `status` slot of the chunk in the goal buffer is set to the value 'start'. This causes the agent to start the ELICIT task. The `begin-task` rule thus implements the delay that is imposed by the 'ready-interval-delay' abELICIT agent property; i.e., the delay that occurs before agents begin the task.

```
(P begin-task
    =goal>
        isa                  sensemaking-goal
        status               not-started
        cycle                =time
    =self>
        isa                  self
        ready-interval-delay =time
    ==>
    =goal>
        status               start
)
```

At the present time, the cognitive model that is used to solve the ELICIT problem (see Section 5) does not exploit the self module, and the behavior of ACT-R agents within the ACT-R/CSSC is therefore not comparable to the behavior of agents seen in the experiments by Manso [42] and others. Adapting the behavior of ACT-R agents in accord with the parameter settings available via the self module requires the addition of production rules resembling the one illustrated in Listing 4. This is one of the focus areas of our future cognitive modeling efforts within the context of the ITA program (see Section 7).

## 5. ELICIT COMPUTATIONAL COGNITIVE MODEL

In order to study the performance of agents engaged in a cognitive task, it is necessary to develop cognitive models that enable agents to perform the task. In the case of ACT-R, the aim of cognitive modeling is to develop models that capture at least some of the cognitive processing routines seen in the case of human agents, while simultaneously respecting the various constraints and limitations of the human cognitive system. The current version of the ECCM developed to support the ELICIT task consists of a total of 124 production rules that, together, enable ACT-R agents to interpret factoids and reason over the semantic content of these factoids in order to infer the likely answer to each of the dimensions of the ELICIT problem, i.e., to identify the time of the terrorist attack, the country in which the attack will take place, the target of the attack, and the group responsible for the attack. The actual development of the ECCM involved a process of knowledge analysis and subsequent prototyping of the reasoning process using the CLIPS (C Language Integrated Production System) expert system shell [45]. This process is described in greater detail in Smart and Sycara [43]. It is important to emphasize that this initial model is limited in the sense that it only enables ACT-R agents to solve ELICIT problems (given access to the relevant factoids); it does not implement *all* the cognitive processes that are required for agents to make decisions about (e.g.) when to share information and who to share information with. In addition, as mentioned in Section 4.2.4, the current version of the model does not incorporate rules that adapt agent behavior with respect to the parameter settings seen in the case of simulations with abELICIT agents. These extensions to the cognitive model are the focus of future modeling efforts (see Section 7).

Listing 5: One of 124 productions that enables an ACT-R agent to solve the ELICIT task.

```
(P evaluate-group-suicide-bombing-tactics-3
=goal>
    isa                 sensemaking-goal
    selected-dimension  who
    status              evaluate-property
    selected-object     =object
    target-attribute    is-suicide-bomber
=imaginal>
    isa                 statement
    object              =object
    attribute           is-suicide-bomber
    value               no
=retrieval>
    isa                 statement
    object              attack
    attribute           is-suicide-bombing
    value               yes
==>
+imaginal>
    isa                 statement
    object              =object
    attribute           is-suspect
    value               no
    source              self
=goal>
    status              check-suspect-status
)
```

As with all ACT-R models, the ECCM consists of a number of chunk-types, chunks and production rules. A key element of the cognitive model is the `statement` chunk-type, which is used to represent an agent's knowledge about the situation as described by the ELICIT factoids. The `statement` chunk-type represents a basic fact about the sensemaking situation, which is captured as an `<object, attribute, value>` triple, similar to the triples seen in the Resource Description Framework [44]. The chunk type contains a number of slots, the most of important of which are the `object`, `attribute` and `value` slots. The following slots are associated with the `statement` chunk type:

- **object:** Contains the name of an object that features as part of a situation model.
- **attribute:** Contains the name of an attribute associated with the object named in the 'object' slot.
- **value:** Contains the value of the attribute named in the 'attribute' slot.
- **is-true:** Indicates whether the statement made about the object in question is true (yes) or false (no). At the present time, all statements are assumed to be true.
- **source:** Specifies the source of the statement. This could be the name of a particular agent, the name of a physical information source, such as a sensor, or the name of an information repository. In cases where the statement exists as the result of an inference made by the agent, the value of the slot will be 'self' (this is the default value).
- **confidence:** Indicates the level of confidence the agent has in the truth or falsity of the statement. A value of 100 (default) indicates maximum confidence, whereas a value of 0 indicates no confidence. At present, this slot is not used as part of the reasoning process; however, it could be used in experiments that aim to study the influence of uncertainty on processes at both the individual and collective (team) levels.

Aside from chunk-types, the model contains a total of 124 production rules, one of which is shown in Figure 5. These rules match against the contents of the buffers associated with each of the core ACT-R modules described in Section 2, as well as the custom modules described in Section 4.2. Rule execution is controlled by task state information that is stored in the ACT-R goal buffer (see Section 2). In particular, the process of evaluating and eliminating suspect entities is decomposed into a number of inference steps, which are illustrated in Figure 3 and described in Table 3. These inference steps correspond to particular stages of the reasoning process associated with the ELICIT task, and each one is associated with a particular subset of rules. By recording which inference step is 'active' at any given time, an agent can track its progress and limit the number of rules that can apply. The rule depicted in Listing 5 exemplifies this: the rule features a conditional element that matches to the `status` slot of a `sensemaking-goal` chunk that is contained in the goal buffer. This rule can only be selected for execution when the value of the `attribute` slot corresponds to 'evaluate-property', and this identifies one of the inference steps associated with the larger reasoning process (see Figure 3).

## 6. USING THE ACT-R/CSSC TO RUN SIMULATION EXPERIMENTS

In order to run simulation experiments with the ACT-R/CSSC, an experimenter must first load all of the Lisp source code files associated with the ACT-R/CSSC into the ACT-R environment¶. Following this, a separate lisp file containing all the configuration instructions for a particular experiment needs to be loaded. This file will typically contain functions that specify the structure of the communication network, the particular factoidset to use, the websites that particular agents are able to access, the timing of each factoid distribution wave, and so on. When the simulation experiment is finally run,

¶The ACT-R environment can be downloaded for free from the ACT-R website [23].

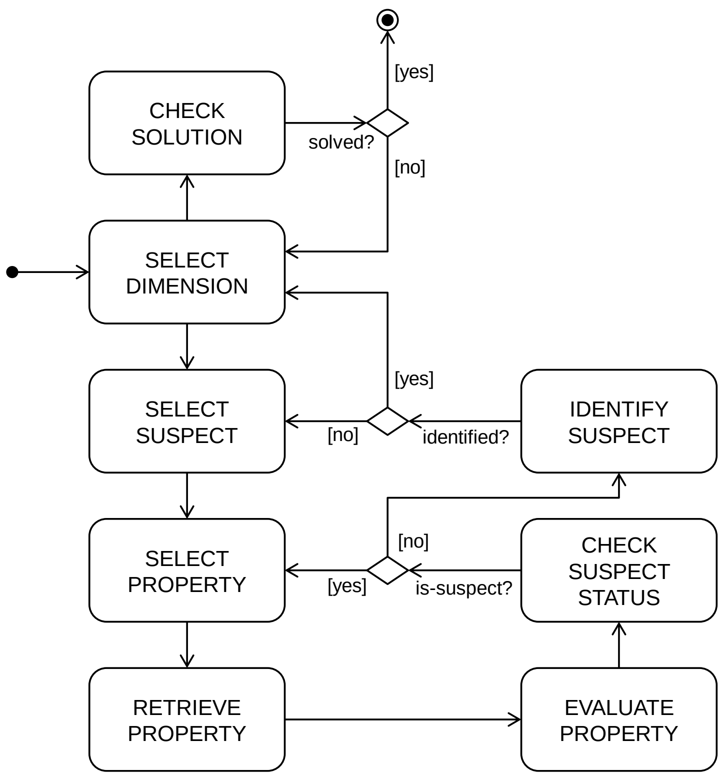


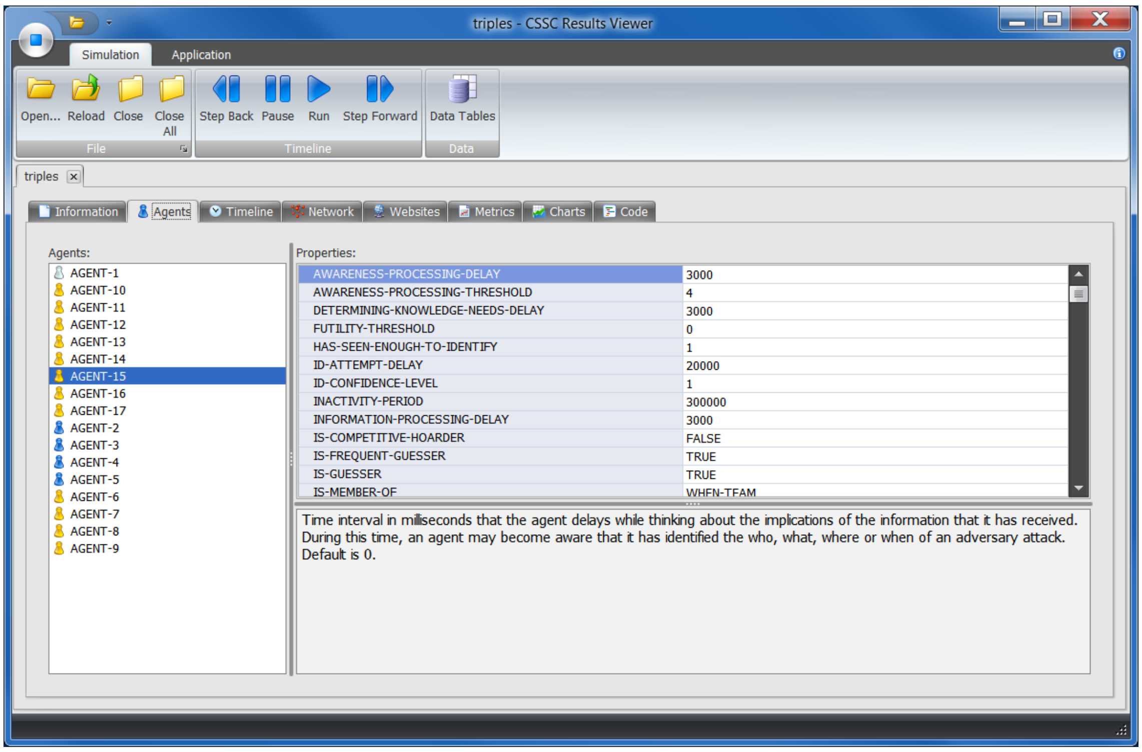
Figure 3: Activity diagram showing the inference steps implemented by an ACT-R agent engaged in the ELICIT task.

Table 3: Stages of the reasoning process that are implemented by the ECCM.

| **Name** | **Description** |
|---|---|
| CHECK-SOLUTION | Determines whether the ELICIT problem has been solved, i.e., whether the who, what, when and where dimensions of the task have been resolved. |
| SELECT-DIMENSION | Selects one of the dimensions (who, what, when or where) of the task to focus on. |
| SELECT-SUSPECT | Selects a particular suspect entity (e.g., a particular group) to focus on. |
| SELECT-PROPERTY | Selects a particular property (e.g., 'is-suicide-bomber') of the selected suspect entity to evaluate. |
| RETRIEVE-PROPERTY | Retrieves information about the selected property from declarative memory. |
| EVALUATE-PROPERTY | Evaluates the information retrieved from memory and assesses whether the suspect status of the selected entity should be modified. |
| CHECK-SUSPECT-STATUS | Determines whether the selected suspect entity is still a suspect following evaluation of the property retrieved in the 'RETRIEVE-PROPERTY' inference step. |
| IDENTIFY-SUSPECT | Assesses whether a particular suspect entity (e.g., a particular group) for a particular dimension (e.g., who dimension) can be identified. If so, the agent will proceed to select a different dimension to focus on. If not, another suspect entity will be selected until all suspect entities of a particular type have been evaluated. At this time, control passes back to the 'SELECT-DIMENSION' inference step. |

the ACT-R/CSSC creates the required number of ACT-R agents and associates each agent with the appropriate cognitive model (e.g., the ECCM). The models are then executed in parallel by the ACT-R environment, and the production rules of each model work to enable an ACT-R agent to interpret messages and reason over their contents. The simulation continues until all the agents have solved the problem or a specified time interval has elapsed. At the conclusion of the experiment, the contents of all the ACT-R/CSSC databases are serialized to text files in a specified directory on the host machine.

The text files that are generated by the ACT-R/CSSC can be loaded by a Windows desktop application, called the



Agents:

AGENT-1
AGENT-10
AGENT-11
AGENT-12
AGENT-13
AGENT-14
AGENT-15
AGENT-16
AGENT-17
AGENT-2
AGENT-3
AGENT-4
AGENT-5
AGENT-6
AGENT-7
AGENT-8
AGENT-9

Properties:

| | |
|---|---|
| AWARENESS-PROCESSING-DELAY | 3000 |
| AWARENESS-PROCESSING-THRESHOLD | 4 |
| DETERMINING-KNOWLEDGE-NEEDS-DELAY | 3000 |
| FUTILITY-THRESHOLD | 0 |
| HAS-SEEN-ENOUGH-TO-IDENTIFY | 1 |
| ID-ATTEMPT-DELAY | 20000 |
| ID-CONFIDENCE-LEVEL | 1 |
| INACTIVITY-PERIOD | 300000 |
| INFORMATION-PROCESSING-DELAY | 3000 |
| IS-COMPETITIVE-HOARDER | FALSE |
| IS-FREQUENT-GUESSER | TRUE |
| IS-GUESSER | TRUE |
| IS-MEMBER-OF | WHEN-TEAM |

Time interval in milliseconds that the agent delays while thinking about the implications of the information that it has received. During this time, an agent may become aware that it has identified the who, what, where or when of an adversary attack. Default is 0.

Figure 4: The 'Agent Tab' of the ACT-R/CSSC Results Viewer.

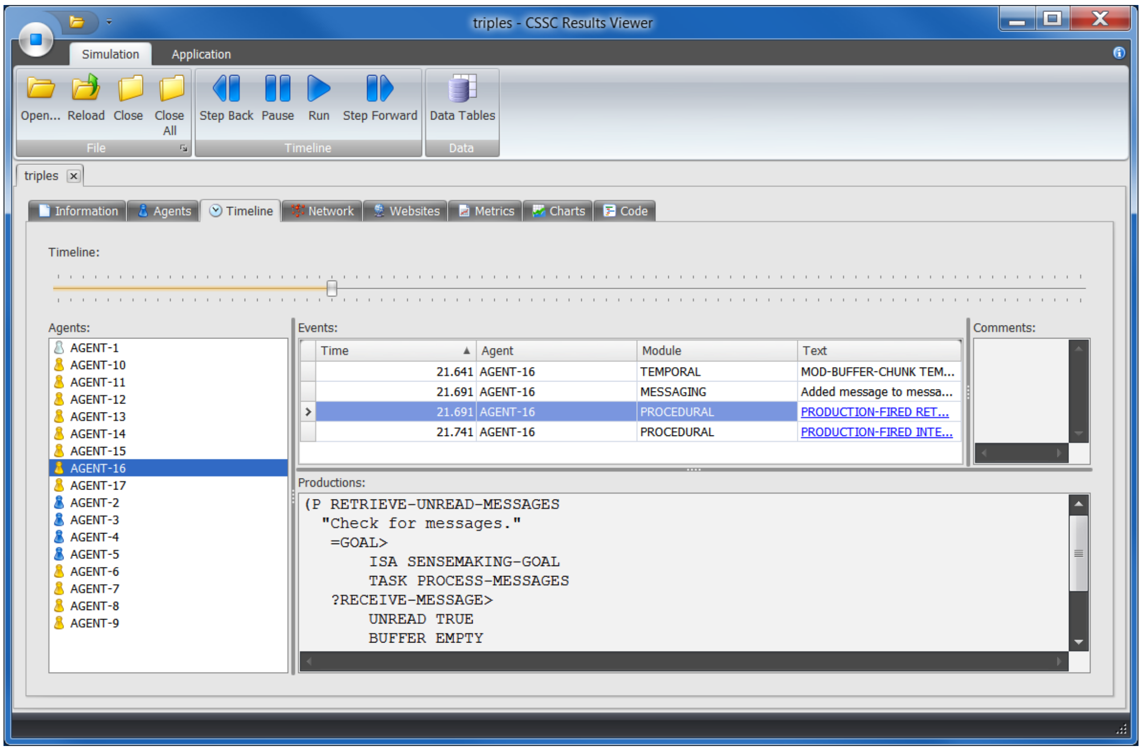
ACT-R/CSSC Results Viewer. This application was developed to support the visualization and analysis of simulation results. The ACT-R/CSSC Results Viewer provides access to all the data generated by the simulation experiment, as well as information about the nature of the experiment that was run. Such information is presented in a series of tabs. Figure 4, for example, shows the 'Agents Tab' of the application. This provides access to all of the agent configuration properties that are accessible to agents via the self module discussed in Section 4.2.4. Another tab of interest is the 'Timeline Tab' (see Figure 5). This tab displays (among other things) all the rules that were executed by particular agents at each step of the simulation, and it therefore enables an experimenter to gain a better understanding of the behavior of agents and the reasons for their actions. Other tabs provide access to information concerning the structure of the communication network ('Network Tab'), the accessibility of particular websites to particular agents ('Websites Tab'), and the value of specific metrics that are typically encountered in the context of ELICIT experiments ('Metrics Tab'), for example, the cognitive self-synchronization values reported by Manso [42].

## 7. FUTURE WORK

The focus of our future research and development efforts within the ITA program fall within a number of areas. Firstly, as mentioned in Section 4.2.4, we aim to extend the ECCM in order to adapt agent behavior with respect to the configuration parameters seen in abELICIT experiments. This will enable us to run experiments similar to those performed with abELICIT agents, and we can therefore compare the performance profiles of both ACT-R agents and abELICIT agents under similar experimental conditions. A specific aim here is to extend the functionality of the existing ACT-R/CSSC and ECCM implementatins to the point where we are able to replicate the experimental conditions described by Manso and Ruddy [11]. Since Manso and Ruddy present results from both human subjects and abELICIT agents, we should be able to compare the performance profiles of ACT-R agents with those obtained from both human subjects and abELICIT agents.

A second area of work relates to the need to run experiments that feature the participation of both human subjects and ACT-R agents. In particular, we wish to extend the ACT-R/CSSC so that human subjects can work alongside ACT-R agents (as part of the same team) in solving the ELICIT problem. This particular capability requires us to implement a human computer interface to support the visualization of factoids. It also requires us to implement features that enable human participants to send messages to other agents (either human or synthetic), post messages to websites and pull messages from websites. These interfaces obviously need to be linked into the ACT-R/CSSC environment, which we hope to accomplish using a network socket-based solution.

Finally, in order to provide a suitable model of human performance in the ELICIT task, we will need to duplicate the kinds of support provided by aspects of the ELICIT task environment, particularly the human-computer interfaces. As



| Time | Agent | Module | Text |
|---|---|---|---|
| 21.641 | AGENT-16 | TEMPORAL | MOD-BUFFER-CHUNK TEM... |
| 21.691 | AGENT-16 | MESSAGING | Added message to messa... |
| 21.691 | AGENT-16 | PROCEDURAL | PRODUCTION-FIRED RET... |
| 21.741 | AGENT-16 | PROCEDURAL | PRODUCTION-FIRED INTE... |

```
(P RETRIEVE-UNREAD-MESSAGES
  "Check for messages."
   =GOAL>
       ISA SENSEMAKING-GOAL
       TASK PROCESS-MESSAGES
   ?RECEIVE-MESSAGE>
       UNREAD TRUE
       BUFFER EMPTY
```

Figure 5: The 'Timeline Tab' of the ACT-R/CSSC Results Viewer.

noted by Smart and Sycara [43], the material elements of a task environment may play a role in influencing cognitive processes at both the individual and collective levels. When it comes to situations where human subjects are attempting to perform the ELICIT task using computer interfaces, the features of the interface may help to alleviate the cognitive burden placed upon subjects in solving the problem; for example, human subjects may be able to use the external display as a form of extended memory (see [46]) for whatever factoids have been received. In the absence of this kind of support, ACT-R agents are required to store all factoids in their declarative memory modules (the equivalent of bio-memory), and this imposes a greater cognitive burden on ACT-R agents relative to their human counterparts. In order to address these sorts of concerns, it will be necessary to develop computational analogues to the task environment seen in the case of human subjects and use 'embodied ACT-R agents' (see [40]) that can interact with these environments in the same way as human subjects. Recent versions of ACT-R provide support for modeling agent-world interactions through the use of what is called a 'device' object. The use of this object to represent features of the task environment within which agents are situated constitutes an important focus area for future research attention.

## 8. CONCLUSION

The aim of cognitive social simulation is to improve our understanding of the complex inter-play between factors that are spread across the cognitive, social and technological domains. This makes cognitive social simulation techniques particularly appealing as a means to undertake experiments into socially-distributed cognition. The current paper reports on the results of an ongoing effort to develop a cognitive social simulation capability that can be used to undertake studies into team cognition using the ACT-R cognitive architecture. In order to test the simulation capability, a computational cognitive model has been developed based on a team-based problem solving task previously used as part of the ELICIT experimentation framework. The cognitive social simulation capability extends the functionality of the core ACT-R implementation using a combination of custom modules and memory-resident databases. These extensions support the ability to run experimental simulations resembling those undertaken in previous studies with the ELICIT task. By comparing the results obtained with the ACT-R simulation capability with those obtained using the ELICIT experimentation platform, it should be possible to evaluate the extent to which ACT-R agents exhibit performance profiles similar to those of their human counterparts. This will help us to assess the extent to which the simulation capability and associated cognitive models can be used to generate findings of predictive relevance to future studies using the ELICIT experimentation framework.

## ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

## REFERENCES

[1] N. J. Cooke, J. C. Gorman, C. W. Myers, and J. L. Duran, "Interactive team cognition," Cognitive Science, vol. 37, no. 2, 2013, pp. 255–285.

[2] G. R. Semin and E. R. Smith, "Socially situated cognition in perspective," Social Cognition, vol. 31, no. 2, 2013, pp. 125–146.

[3] P. R. Smart and K. Sycara, "Collective sensemaking and military coalitions," Intelligent Systems, vol. 28, no. 1, 2013, pp. 50–56.

[4] G. Theiner, C. Allen, and R. L. Goldstone, "Recognizing group cognition," Cognitive Systems Research, vol. 11, 2010, pp. 378–395.

[5] M. Kearns, "Experiments in social computation," Communications of the ACM, vol. 55, no. 10, 2012, pp. 56–67.

[6] J. Sutton, C. Harris, P. Keil, and A. Barnier, "The psychology of memory, extended cognition, and socially distributed remembering," Phenomenology and the Cognitive Sciences, vol. 9, no. 4, 2010, pp. 521–560.

[7] V. Hinsz, R. Tindale, and D. Vollrath, "The emerging conceptualization of groups as information processors," Psychological Bulletin, vol. 121, 1997, pp. 43–64.

[8] N. J. Cooke, J. C. Gorman, and W. J. L, "Team cognition," in Handbook of Applied Cognition, 2nd ed., F. T. Durso, R. S. Nickerson, S. T. Dumais, S. Lewandowsky, and T. J. Perfect, Eds. Chichester, UK: John Wiley & Sons, 2007.

[9] E. Hutchins, Cognition in the Wild. Cambridge, Massachusetts, USA: MIT Press, 1995.

[10] I. Couzin, "Collective cognition in animal groups," Trends In Cognitive Sciences, vol. 13, no. 1, 2009, pp. 36–43.

[11] M. Manso and M. Ruddy, "Comparison between human and agent runs in the ELICIT N2C2M2 validation experiments," in 18th International Command and Control Research and Technology Symposium (ICCRTS), Alexandria, Virginia, USA, 2013.

[12] M. Kang, "The effects of agent activeness and cooperativeness on team decision efficiency: A computational simulation study using Team-Soar," International Journal of Human-Computer Studies, vol. 65, no. 6, 2007, pp. 497–510.

[13] R. Sun and I. Naveh, "Simulating organizational decision-making using a cognitively realistic agent model," Journal of Artifical Societies and Social Simulation, vol. 7, no. 3, 2004.

[14] M. Prietula, K. Carley, and L. Gasser, Simulating Organizations: Computational Models of Institutions and Groups. Boston, Massachusetts, USA: The MIT Press, 1998.

[15] "SWARM - Summary," URL: http://savannah.nongnu.org/projects/swarm [accessed: 2014-04-21].

[16] "Repast Suite," URL: http://repast.sourceforge.net/ [accessed: 2014-04-21].

[17] R. Sun, "Cognitive social simulation incorporating cognitive architectures," Intelligent Systems, vol. 22, no. 5, 2007, pp. 33–39.

[18] P. Thagard, "Cognitive architectures," in The Cambridge Handbook of Cognitive Science, K. Frankish and W. M. Ramsey, Eds. Cambridge, UK: Cambridge University Press, 2012, pp. 50–70.

[19] J. R. Anderson, How Can the Human Mind Occur in the Physical Universe? Oxford, UK: Oxford University Press, 2007.

[20] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," Psychological Review, vol. 111, no. 4, 2004, pp. 1036–1060.

[21] D. Reitter and C. Lebiere, "Social cognition: Memory decay and adaptive information filtering for robust information maintenance," in 26th AAAI Conference on Artificial Intelligence, Toronto, Canada, 2012.

[22] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," Cognitive Systems Research, vol. 10, no. 2, 2009, pp. 141–160.

[23] "ACT-R," URL: http://act-r.psy.cmu.edu/ [accessed: 2014-04-20].

[24] J. E. Morrison, "A review of computer-based human behavior representations and their relation to military simulations," Institute for Defense Analyses, Tech. Rep. IDA Paper P-3845, 2003.

[25] D. D. Salvucci, "Modeling driver behavior in a cognitive architecture," Human Factors, vol. 48, no. 2, 2006, pp. 362–380.

[26] D. D. Salvucci and K. L. Macuga, "Predicting the effects of cellular-phone dialing on driver performance," Cognitive Systems Research, vol. 3, no. 1, 2002, pp. 95–102.

[27] G. Gunzelmann, R. L. Moore, D. D. Salvucci, and K. A. Gluck, "Sleep loss and driver performance: Quantitative predictions with zero free parameters," Cognitive Systems Research, vol. 12, no. 2, 2011, pp. 154–163.

[28] J. R. Anderson, Y. Qin, K.-J. Jung, and C. S. Carter, "Information-processing modules and their relative modality specificity," Cognitive Psychology, vol. 54, no. 3, 2007, pp. 185–217.

[29] S. M. Rodgers, C. W. Myers, J. Ball, and M. D. Freiman, "Toward a situation model in a cognitive architecture," Computational and Mathematical Organization Theory, vol. 19, no. 3, 2013, pp. 313–345.

[30] J. E. Laird, A. Newell, and P. S. Rosenbloom, "SOAR: An architecture for general intelligence," Artificial Intelligence, vol. 33, no. 1, 1987, pp. 1–64.

[31] P. Rosenbloom, J. Laird, A. Newell, and R. McCarl, "A preliminary analysis of the SOAR architecture as a basis for general intelligence," in Foundations of Artificial Intelligence, D. Kirsh, Ed. Cambridge, Massachusetts: MIT Press, 1992, pp. 289–326.

[32] J. E. Laird, The SOAR Cognitive Architecture. Boston, Massachusetts, USA: MIT Press, 2012.

[33] R. Sun, "The CLARION cognitive architecture: Extending cognitive modeling to social simulation," in Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Interaction, R. Sun, Ed. New York, New York, USA: Cambridge University Press, 2006, pp. 79–99.

[34] B. J. Best and C. Lebiere, "Teamwork, communication, and planning in ACT-R agents engaging in urban combat in virtual environments," in Workshop on Cognitive Modeling of Agents and Multi-Agent Interactions, Acapulco, Mexico, 2003.

[35] J. Ball, C. Myers, A. Heiberg, N. J. Cooke, M. Matessa, M. Freiman, and S. Rodgers, "The synthetic teammate project," Computational and Mathematical Organization Theory, vol. 16, no. 3, 2010, pp. 271–299.

[36] M. Ruddy, "ELICIT – the experimental laboratory for investigating collaboration, information sharing and trust," in 12th International Command and Control Research and Technology Symposium (ICCRTS), Newport, Rhode Island, USA, 2007.

[37] M. Ruddy, D. M. Wynn, and J. McEver, "Instantiation of a sensemaking agent for use with ELICIT experimentation," in 14th International Command and Control Research and Technology Symposium, Washington D.C., USA, 2009.

[38] D. M. Wynn, M. Ruddy, and M. E. Nissen, "Command & control in virtual environments: Tailoring software agents to emulate specific people," in 15th International Command and Control Research and Technology Symposium, Santa Monica, California, USA, 2010.

[39] P. R. Smart, K. Sycara, and Y. Tang, "Using cognitive architectures to study issues in team cognition in a complex task environment," in SPIE Defense, Security and Sensing, Baltimore, Maryland, USA, 2014.

[40] M. J. Schoelles and W. D. Gray, "Argus: A suite of tools for research in complex cognition," Behavior Research Methods, Instruments, & Computers, vol. 33, no. 2, 2001, pp. 130–140.

[41] M. J. Schoelles, H. Neth, C. W. Myers, and W. D. Gray, "Steps towards integrated models of cognitive systems: A levels-of-analysis approach to comparing human performance to model predictions in a complex task environment," in 28th Annual Conference of the Cognitive Science Society, Vancouver, British Columbia, Canada, 2006.

[42] M. Manso, "N2C2M2 validation using abELICIT: Design and analysis of ELICIT runs using software agents," in 17th International Command and Control Research and Technology Symposium, Fairfax, Virgina, USA, 2012.

[43] P. R. Smart and K. Sycara, "Cognitive social simulation and collective sensemaking: An approach using the ACT-R cognitive architecture," in 6th International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE'14), Venice, Italy, 2014.

[44] "RDF – Semantic Web Standards," URL: http://www.w3.org/RDF/ [accessed: 2014-04-14].

[45] "CLIPS: A Tool for Building Expert Systems," URL: http://clipsrules.sourceforge.net/ [accessed: 2014-04-14].

[46] P. R. Smart, "Extended memory, the extended mind, and the nature of technology-mediated memory enhancement," in 1st ITA Workshop on Network-Enabled Cognition: The Contribution of Social and Technological Networks to Human Cognition, Maryland, USA, 2010.

## APPENDIX A. ACT-R/CSSC DATABASES

Table 4: Structure of the Lisp databases used in the ACT-R/CSSC.

| **Field Name** | **Description** |
|---|---|
| *Message Database* | |
| Source | Specifies the agent that sent the message. |
| Target | Specifies the intended recipient of the message. |
| Text | Specifies the text of the message. |
| Factoid Number | If the message represents an ELICIT factoid, this value specifies the number of the factoid. |
| Read Status | Indicates whether the message has been processed by the intended recipient. |
| Timestamp | Records the time the message was sent by the originating agent. |
| *Factoid Database* | |
| Factoidset | Specifies the name of the factoidset to which a particular factoid belongs. |
| Factoid Number | Specifies the number of the factoid within the relevant factoidset. |
| Impact | Specifies the type of the factoid, i.e., whether the factoid is a key, expert, supportive or noise factoid. |
| Factoid Type | Specifies the dimension (i.e., who, what, where, when) of the ELICIT problem that the factoid is relevant to. |
| Role Number | Specifies the role number of the agent that the factoid should be distributed to in the context of a distribution wave. |
| Distribution Wave | Specifies the number of the distribution wave that the factoid should be distributed within. |
| Type Impact Count | Specifies the number of the factoid within both its impact and type categories. |
| Target Agent | Specifies the name of the agent that the factoid should be distributed to in the context of a distribution wave. |
| *Triple Database* | |
| Subject | Specifies the subject component of the triple. |
| Predicate | Specifies the predicate component of the triple. |
| Object | Specifies the object component of the triple. |
| *Web Message Database* | |
| Source | Specifies the agent that posted the message to the website. |
| Text | Specifies the text of the message. |
| Factoid Number | If the message represents an ELICIT factoid, this value specifies the number of the factoid. |
| Timestamp | Records the time the message was posted to the website. |
| Website | Specifies the ELICIT website that the message was posted to. |
| *Action Database* | |
| Action | Specifies the type of action that was performed, e.g., pull-action. |
| Agent | Specifies the agent that performed the action. |
| Target | Specifies the target of the action, such as a particular website or agent. |
| Object | Specifies an object that was involved in the action, e.g., a particular factoid. |
| Timestamp | Records the time the action was made. |
| *Identification Database* | |
| Source | Specifies the agent that made the identification attempt. |
| Who Suspect | Specifies the suspect entity that was identified for the 'who' dimension of the problem. |
| What Suspect | Specifies the suspect entity that was identified for the 'what' dimension of the problem. |
| Where Suspect | Specifies the suspect entity that was identified for the 'where' dimension of the problem. |
| When Suspect | Specifies the suspect entity that was identified for the 'when' dimension of the problem. |
| Timestamp | Records the time the identification attempt was made. |
| Status | Specifies whether the identification was correct, partially correct, or incorrect. |
| *Connection Database* | |
| Source | Specifies the agent that can send messages to the target agent. |
| Target | Specifies the agent that can receive messages sent by the source agent. |