Human-Agent Teamwork for Cyber Sensemaking in Network Operations

Jeffrey M. Bradshaw, Larry Bunch, Tom Eskridge, Paul J. Feltovich
Robert R. Hoffman, Matthew Johnson, James Lott
Florida Institute for Human and Machine Cognition (IHMC)
Pensacola, FL
{jbradshaw, lbunch, teskridge, pfeltovich, rhoffman, mjohnson, jlott}@ihmc.us

Marco M. Carvalho
Florida Institute of Technology (FIT)
Melbourne, FL
mcarvalho@fit.edu

ABSTRACT

In this article, we describe how we implement human-machine teamwork in Sol, a framework for cyber operations [3]. Specifically, we describe how the use of software agents (*Luna*), semantically rich policies (*KAoS*), and principles of visualization grounded in an understanding of human perception and cognition (*OZ)* can be used to support distributed sensemaking and effective response to cyber threats.

1. INTRODUCTION

Despite the significant attention being given to the critical challenges of cyber operations, the ability to keep up with the increasing volume and sophistication of network attacks is seriously lagging. To better empower network professionals, we need to seriously rethink the way cyber operations tools and approaches have been conceived, developed, and deployed.

2. AGENTS, POLICIES, AND VISUALIZATION IN SOL

In this section, we introduce the core technical capabilities on which Sol relies: the *Luna Agent Framework, KAoS Policy Services Framework*, and *OZ Visualization Principles.*

2.1 Luna Agent Framework

To meet the challenging demands of cyber operations, we have developed a new agent framework called *Luna*, named for the founder of Pensacola, Tristán de Luna y Arellano (1519 – 1571) [4]. Three reasons dictated our decision to implement Luna.

First, in considering the *security requirements* of current software agent platforms, the key role of policy constraints that could govern behavior at every level of the agent system readily became apparent (see section 2.2 below). We have found that the ability to use policy to impose constraints on agent behavior, in essence providing a guarantee to people that agent autonomy would always be exercised within specified bounds, gave people the assurance they needed to feel that highly capable agents could act in a trustworthy, predictable, and safe manner.

Second, with respect to the need for a platform supporting the interactive *formulation of common agent tasks by end users*, rather than by software developers, we believe that policy systems may also prove useful. In past experience, we have learned that many common tasks can be formulated as declarative obligation policies that require given actions when triggered by a specified context.

Finally, to satisfy the need for a platform that would provide built-in support for effective *coordination of joint activity within mixed teams of humans and agents*, we believe that a policy-based approach also provides a viable option. Based on our research and development experience in a variety of applications involving the coordination of human-agent-robot teamwork (HART), we believe that important aspects of teamwork can be supported by policy-based mechanisms [2].

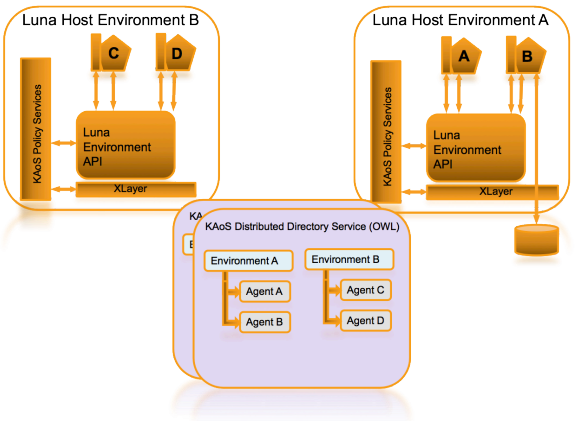

Figure 1. Luna Conceptual Architecture.

Figure 1 shows conceptually how KAoS integrates with Luna to provide services and to enforce policies. An OWL representation of Luna is maintained within the KAoS Distributed Directory Service. Through its interactions with the Luna host environment, KAoS regulates the lifecycle of both the environment (e.g., start and stop Luna) and the agents (e.g., create, pause, resume, stop, and move agents). Policy can also regulate environment context for shared agent memory (e.g., getting and setting its properties), allowing efficient parallel processing of large data sets. An agent-based implementation of context mirroring across different Luna environments is provided. Through policy, the Luna host environment also governs agent progress appraisal [7]—allowing analysts to easily learn what tasks and which agents are significantly ahead or behind schedule, and thus replanning their own efforts on interdependent tasks accordingly.

In order to support dynamic scalability, load balancing, adaptive resource management, and specific application needs, the Luna platform supports the policy-governed option of allowing the *state* of agents (vs. *code* of agents) to migrate between operating environments and hosts. The Luna environment maintains agent mailboxes with message forwarding when agents migrate. Luna state mobility will provide the foundation for future implementation of agent persistence (i.e., saving and loading agent state to a persistent store).

One of the most important innovations in Luna is the ability to add custom agent actions to the policy ontology, based on their Java implementation, even when source code is not available. We provide a Java2OWL tool to automate this task.

2.2 KAoS Policy Services Framework

Because agents are powerful, we use powerful policy management and enforcement frameworks to govern their actions. Whereas many special-purpose policy approaches are optimized only for specific kinds of tasks (e.g., access control), the ontology-based approach of KAoS enables semantically-rich specifications of virtually any kind of constraint on any specific kind of action in richly defined dynamic contexts. KAoS supports not only the ability to permit or forbid an action in a given context, but also to require that certain actions be performed when a dynamic, context-specific trigger is activated (e.g., start doing X, stop doing Y, reduce your bandwidth usage, log certain actions)—or to waive such an obligation dynamically if the situation warrants.

The KAoS Policy Services framework [10] was the first to offer an ontology-based approach (based on the W3C standard, OWL 2 (http://www.w3.org/TR/owl2-overview/) to policy representation and reasoning. It is currently the most successful and mature of all such efforts. Following collaborative efforts by the NSA-sponsored Digital Policy Management (DPM) Architecture Group and IHMC, the KAoS core ontology was adopted as the basis for future standards efforts in DPM.

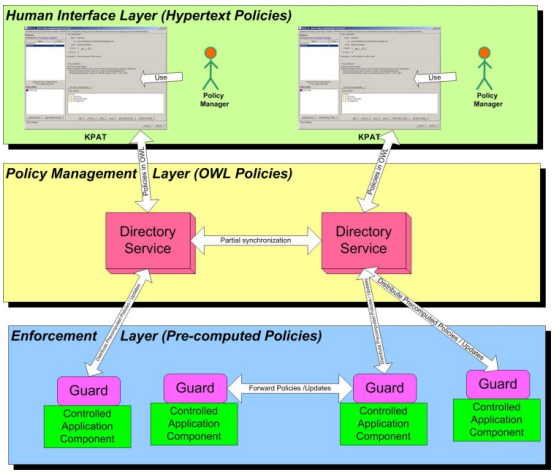

Figure 2. KAoS Conceptual Architecture**.**

The basic elements of the KAoS architecture are shown in Figure 2. The three layers of functionality correspond to three different policy representations:

- *Human Interface Layer*: This layer uses a hypertext-like graphical interface (KAoS Policy Administration Tool — KPAT) for policy specification in the form of constrained English sentences. The vocabulary is automatically provided from the relevant core ontologies or application-specific ones. Besides KPAT's use in policy specification and analysis, it is employed for administrative tasks such as browsing and loading ontologies, and domain and Guard management. The generic KPAT interface can be easily customized or replaced.
- *Policy Management Layer*: Within this layer, OWL is used to encode and manage policy-related information. The KAoS Distributed Directory Service (DDS) encapsulates a set of ontology reasoning mechanisms over the policies, used for policy deconfliction, analysis, and testing.
- *Policy Monitoring and Enforcement Layer*: KAoS automatically "compiles" OWL policies to a very efficient format that can be used for monitoring and enforcement. This representation provides the grounding for abstract ontology terms, connecting them to the instances in the runtime environment and to other policy-related information (e.g., dynamic state or history). Extensibility is supported through a framework with well-defined interfaces that can be enriched to support new kinds of policies. KAoS Guards residing in this layer are integrated with the controlled application and provide an API for policy information querying and decision-making.

KAoS is integrated into IHMC's Luna agent framework, as well as third-party agent platforms and traditional service-oriented architectures. Preliminary work has been done on agent learning mechanisms that propagate learning with localized opportunistic mechanisms inspired by biological analogues. In addition, we are developing capabilities for KAoS to take advantage of localized agent learning results by allowing new policies to be constructed programmatically, with optional human oversight. This would allow learning results from groups of individual agents that are of high generality or urgency to be rapidly propagated to whole classes of other agents.

In support of human-agent teamwork, each Luna agent is governed by policies designed to assure its observability (e.g., mandatory status updates at an appropriate frequency, or in response to specified events), directability (e.g., immediate responsiveness to redirection due to policy changes), interpredictability (e.g., obligation policies assuring that required behavior will be executed within a specified time period), adaptation (e.g., policies governing the range of adaptations permitted and the process of propagation to other agents), support for multiplicity (e.g., policies governing synchronization of multiple perspectives), and trustworthiness (e.g., policies assuring the observability of parameters indicating the reliability of agent operations). In addition, KAoS policies also help with resilience, ensuring that the entire system adapts automatically to changes in context, environment, task reprioritization, or resources. New or modified policies can be made effective immediately at runtime.

2.3 OZ Visualization Principles

Our approach to real-time cyber sensemaking displays is informed by lessons learned in the design of IHMC's highly-successful OZ flight display [9]. Instead of relying on a continual visual scan of cockpit instruments, as on the traditional flight display, OZ presents information holistically and in the context of the current state of the world outside. Presenting flight performance information *in context* allows people to more easily maintain overall situation awareness. Presenting information *holistically* allows dependencies among key flight parameters to be made salient through the direct perception of visual primitives. Modifications made to any part of the model through pilot input or changes in the operating environment immediately affect all related visual elements so the operator implicitly learns deep model relationships in context.

Though the display's reliance on colored lines and dots on a black background may seem a primitive throwback to first-generation video games, this simplicity is by design, based on a sophisticated understanding of the latest research results in human perception and cognition. Instead of relying on the slow and small human focal vision system, OZ is designed to use the fast and robust *ambient vision system*—the same system that people use to quickly and successfully navigate crowded hallways without conscious thought or to catch a football on the run. As another example, OZ exploits the capabilities of human vision for quickly perceiving changes by using movement to convey difficult, correlated information.

Due to these and other features, experimentation has repeatedly demonstrated the superiority of OZ over traditional displays in minimizing pilot error, reducing pilot disorientation, and maintaining situation awareness. Because of the OZ display's reliance on the ambient visual system, its advantages are shown even more dramatically in experimental conditions where the pilot is temporarily blinded by a flash of light (as when, e.g., impaired by lack of oxygen) or distracted by performing auxiliary visual tasks that rely on the focal vision system (e.g., reading). Beyond its role in simplifying flight-related tasks, the integrated performance model has an added training benefit—helicopter pilots trained using the hover functionality of OZ are able to more quickly acquire the depth of understanding necessary to master difficult challenges unique to rotorcraft flight, and fixed-wing pilots learn faster, retain training longer, and have a deeper understanding of the fundamental rules of flight than their conventionally-trained counterparts.

We apply these innovations in interactive visualization, and other domain-specific features, in the development of visual displays for complex real-time data in cyber operations applications.

3. HUMAN-AGENT TEAMWORK IN SOL

In this section, we outline our general approach to human-agent teamwork in Sol, emphasizing how we apply the concepts of *coactive emergence* in the design of work methods for distributed sensemaking in cyber operations and of *polycentric governance* in engineering for resilience of the joint human-machine system. Then we give specific examples of how this is done in practice.

3.1 Coactive Emergence

We characterize our approach to human-agent teamwork by the term *coactive emergence*. It describes a continuous iterative process whereby useful interpretations of data are developed, host and network configurations are adjusted, and effective responses to threats are undertaken through the interplay of joint sensemaking, decision-making, and task execution activities performed by analysts and software agents in tandem [3].

The word "coactive" emphasizes the joint, simultaneous, and interdependent nature of such collaboration among analysts and agents. Figure 3 illustrates how this applies in Sol: 1) Agents are pre-coded in Java to perform particular classes of analytic tasks. Analysts manage the work of software agents through policy constraints that direct their sensemaking and task execution activities; 2) Policy-governed agents work together to interpret real-time data and to manipulate host and network configurations, optionally enriching their capabilities through machine learning techniques; 3) Agents work together to enrich their findings with additional information gleaned through learning (e.g., hypothesized correlations between data sets of interest, anticipated future trends); 4) Agents may aggregate and present their findings by visually annotating graphical displays in real-time in order to highlight and draw the attention of the analyst to anomalous or otherwise interesting elements, such as possible attacks. Analysts interact with these displays in order monitor ongoing progress and effectiveness, and to explore and evaluate hypotheses and options; 5) As agent-derived information is presented to analysts, they may agree or disagree with agent findings, leading to further corrections and refinements of interpretations, and consideration of response options; 6) Analysts continue to direct and redirect ongoing agent activity through the construction of new agents, modification of agent policies, and extensions to lines of inquiry. We are also developing methods for system "hardening" in real time by empowering agents to change any number of configurations in near real-time through policy [6].

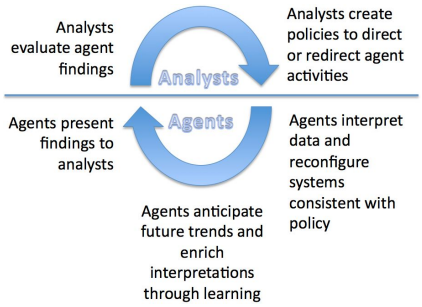

Figure 3. The Coactive Emergence Cycle

3.2 Polycentric Governance

Within the framework of resilient systems engineering, Woods and Branlat have discussed important patterns that lead to failure in complex systems [11]. We are developing agent-based methods to provide support for adaptive performance in the face of stressors and surprise through the principles of polycentric governance [8].

A related notion of organic resilience [5] relies heavily on biologically-inspired analogues and self-organizing strategies for the management and defense of distributed complex systems. As with many biological systems, the goal of an organic resilience approach is to, as much as possible, avoid static and centralized single-point-of-failure solutions for organizing. Thus, although groups of agents within the system are collectively responsible for jointly executing various tasks, the specific responsibilities assigned to agents are not completely sorted out in advance. The goal is to allow the agents to self-organize within the constraints of their individual capabilities, the current applicable policies, and current availability of agents. Applied to organic resilience, policy-based collective obligations provide the regulatory mechanisms that enable effective and coactive coordination algorithms.

3.3 Agent-Based Processing and Tagging

Agents play a variety of roles in Sol. Among the most demanding is in multi-layer agent processing and tagging of live or retrospectively played-back NetFlow data representing worldwide Internet traffic. A high-level view of roles and relationships among agents relating to these functions is shown in Figure 4.

Incoming UDP traffic goes to a NetFlow agent for parsing and transformation into Java objects (1). The NetFlow agent sends the data to any number of Tagger agents that work in parallel in real-time to tag the data (2). For example, Watchlist agents tag data that appears on whitelists or blacklists while IDS Match agents tag data corresponding to intrusion detection alerts. Drawing on selected results from low-level tagging agents, Attack pattern agents may be defined to look for higher-level attack patterns. A system of semaphores ensures that all the Tagger agents have completed their work before the NetFlow agent sends results to the Flow Cache (3). NetFlow Visualization agents enforce policies that mediate data being sent to analyst displays, ensuring, among other things, that data not authorized for viewing by particular users are automatically filtered out (4).

The Esper complex event processor [1] provides an example of Sol support for efficient ad hoc queries of many types that can be initiated and consumed by other visualization agents (e.g., Stripchart View agent) or by agents of other types for further processing (5). We are also considering the use of Esper for data stream handling further upstream in the agent analytic process.

CogLog Correlator agents ingest combined data from selected Tagger agents operating on real-time data (6) and historical data within the CogLog (7). The CogLog is a Semantic-Wiki-based tool prototype with which software agents and human analysts can maintain and use a log of findings pertinent to a given investigation, while also linking to other relevant information from prior cases. Unlike the real-time Tagger agents, the Correlator agent can perform deeper kinds of analytics in "out of band" mode. Among other things, this correlated information can help different analysts "connect the dots" between related investigative efforts. The Correlator agents may send additional data annotations to NetFlow Visualization agents and/or to agents supporting other visualizations (e.g., Connection Graph view) (8). Our Attack Pattern Learning Agents provide another example of an "out of band" agent type. These agents consume and process all NetFlows (rather than just subsets of tagged data produced by Tagger agents) in order to learn and propagate useful threat patterns. Agents can provide active, actionable information by generating "Live Advisories." Remote colleagues can view the rationale for the advisory, replay the relevant data—and, potentially, launch protective actions.

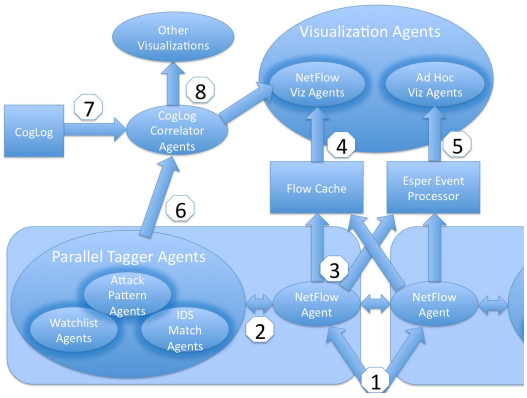

Figure 4. Agent Processing and Tagging of NetFlow Data.

In the future, exploration of larger questions of adversarial intent, attack strategies, and social connections among attackers could also proceed along similar lines of increasing abstraction in agent processing. The ability to reduce perception and reasoning requirements on the analyst through fixed or ad hoc organizations of agents processing visual and logical data dimensions is a major benefit of agent-based analytics.

3.4 The Flow Capacitor

The Flow Capacitor (also known as "Aurora") is an example of a highly-configurable, interactive 3D visualization of Internet traffic based on OZ principles. The input to this visualization is NetFlow records.

A major motivator in the development of this interactive visualization was to be able to show and discriminate among large numbers of NetFlows moving across networks simultaneously in near real-time. In order to do this, the design started with the simplest possible representation for a single network event: a short line segment, or "dart." These darts can be made surprisingly rich (compared to a single point) with multiple colors, length, width, orientation, and glyph annotations. In order to show information about where the dart was coming from and going to, the endpoints of each dart's journey through time is projected onto source and destination 'planes' in 3D space. The period of time represented between the top and bottom planes can be configured to any length, from weeks or days to milliseconds. Other kinds of planes (e.g., treemaps), revealing how different groups of NetFlows cluster when their metadata is projected onto abstract property spaces, can also be defined.

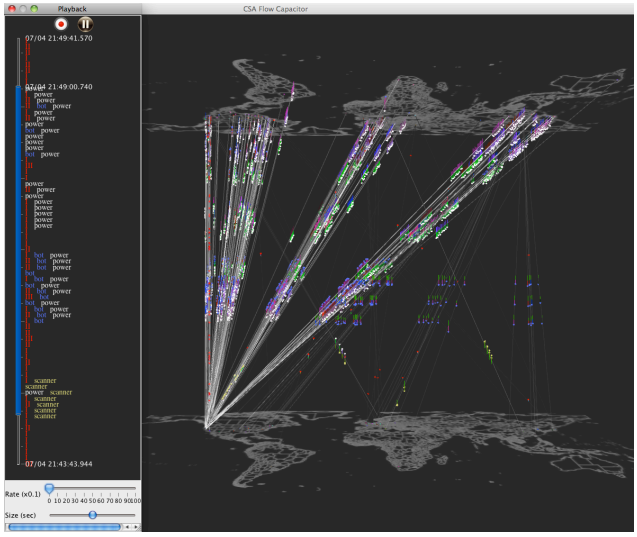

Figure 5. Distributed Denial-of-Service Attack Example.

Analysts visually map selected results of agent tagging relating to data of interest so it can be easily noticed in the visual display. The display is designed so that it could be adapted to show other kinds of events (e.g., financial transactions, travel, spread of diseases, disaster-related information). Any number or kind of planes could be stacked or otherwise arranged topographically to answer questions about complex, high-tempo situations. From a single snapshot of the Flow Capacitor in Figure 5, we can see the unfolding of a sequence of events leading up to a distributed denial-of-service attack portrayed in graphic clarity.

The innovations in human-agent collaboration embodied in Sol suggest significant new directions in automated assistance for cyber operations.

Acknowledgements. Many thanks to our US Department of Defense sponsors for their enthusiastic support of this work.

4. REFERENCES

1. EsperTech. In. http://esper.codehaus.org/. (accessed 18 July, 2012).
2. Bradshaw, J.M., P. Feltovich, and M. Johnson. "Human-Agent Interaction." In *Handbook of Human-Machine Interaction*, edited by G. Boy, 283-302. Ashgate, 2011.
3. Bradshaw, J.M., M. Carvalho, L. Bunch, T. Eskridge, P.J. Feltovich, C. Forsythe, R.R. Hoffman, M. Johnson, D. Kidwell, and D.D. Woods. "Coactive emergence as a sensemaking strategy for cyber operations." Manuscript submitted for publication, 2012.
4. Bunch, L., J.M. Bradshaw, M. Carvalho, T. Eskridge, P.J. Feltovich, J. Lott, and A. Uszok. "Human-Agent Teamwork in Cyber Operations: Supporting Co-Evolution of Tasks and Artifacts with Luna." Presented at the Tenth German Conference on Multiagent System Technologies (MATES 2012) (LNAI 7598), Trier, Germany, October 10-12, 2012, 53-67.
5. Carvalho, M., T. Lamkin, and C. Perez. "Organic resilience for tactical environments." In *Fifth International ICST Confernece on Bio-Inspired Models of Network, Information, and Computing Systems (Bionetics)*. Boston, MA, 2010.
6. Carvalho, M., J.M. Bradshaw, L. Bunch, T. Eskridge, P.J. Feltovich, R.R. Hoffman, and D. Kidwell. "Command and control requirements for Moving Target Defense." *IEEE Intelligent Systems 27*, no. 3 (2012): 79-85.
7. Feltovich, P., J.M. Bradshaw, W.J. Clancey, M. Johnson, and L. Bunch. "Progress appraisal as a challenging element of coordination in human and machine joint activity." Presented at the Engineering Societies for the Agents World VIII, Athens Greece, October, 2007.
8. Ostrom, E. 2008. Polycentric systems as one approach for solving collective-action problems (SSRN-id130469). In *Social Science Research Network*. http://ssrn.com/abstract=1304697. (accessed September 28, 2012).
9. Still, D.L., T. Eskridge, and L.A. Temme. "Interface for non-pilot UAV control." Presented at the Human Factors of UAVs Workshop, Mesa, AZ 2004.
10. Uszok, A., J.M. Bradshaw, J. Lott, M. Johnson, M. Breedy, M. Vignati, K. Whittaker, K. Jakubowski, and J. Bowcock. "Toward a Flexible Ontology-Based Policy Approach for Network Operations Using the KAoS Framework." Presented at the The 2011 Military Communications Conference (MILCOM 2011) 2011, 1108-1114.
11. Woods, D.D. and M. Branlat. "Basic patterns in how complex systems fail." In *Resilience Engineering in Practice*, edited by E. Hollnagel, J. Paries, D.D. Woods, and J. Wreathall, 127-143. Burlington, VT: Ashgate, 2008.