In R. A. Sottilare, A. C. Graesser, X. Hu, A. M. Olney, B. D. Nye, A. M. Sinatra (Eds.). Design Recommendations for Intelligent Tutoring Systems, Volume 4, Domain Modeling (pp. 69–89). Orlando, FL: US Army Research Laboratory, 2016.

CHAPTER 6 – A Work Practice Simulation Approach to Modeling Socio-Technical Domains

Benjamin Bell,[1] William J. Clancey,[2] and Winston Bennett, Jr.[3]
[1]Aqru Research and Technology, [2] Florida Institute for Human and Machine Cognition,
[3] US Air Force Research Laboratory

Overview

A domain model can incorporate a rich and realistic context in a tutoring environment. Domains can be modeled to provide an environment with "real-world" look and feel with a suite of behaviors that create a dynamic context for learning. The construct of "context" is often seen as something objective or completely physical in the world. A domain model for a construction crane simulator may define properties and behaviors of the crane to include its lift capacity and boom length, and will define objects in the world that act upon the crane and vice versa. However, the context of human behavior is also conceptual and subjective. Conceptual context is relational and affected by perceptions, and thus is often dynamic. Domain models should capture how context is conceptually constructed through and during interactions of monitoring, moving in, and manipulating the world.

We adopt a socio-technical approach to domain modeling to adequately capture and model how understanding of context develops in activity, and people's activity within that understanding. Our approach to domain modeling includes a spatial/geographical model, cultural features and artifacts, and information systems, to reflect a broad meaning of "context" that includes what the learner is doing physically and mentally, how the learner is conceiving what the learner is doing (which in more general terms is less defined than tasks), and what the learner is perceiving in the environment. In other words, the context for behavior is always conceptual, and is about a socio-technical, physical world in which the person is an actor.

In this chapter, we present a methodology for modeling domains that employs a socio-technical framework for capturing the activities of individuals and the socio-technical context. We discuss the need for both task analyses and psychosocial activity analyses and how those methods vary for different domains. We describe a computational tool used for building and running these models and summarize several applications of this approach. Finally, we present new work that employs these models for predictive analysis in simulations of uncertain, complex threats, and conclude with implications for domain modeling in intelligent tutoring systems (ITSs).

Introduction

Remarkable advances in artificial intelligence (AI), human-systems integration, and digital communications are providing automation systems of unprecedented sophistication. In this chapter, we use the term AI to describe an agent that can serve one or more humans as an intelligent assistant in either actual systems or in training simulations. An AI onboard an aircraft could, for instance, monitor the aircraft, interpret and carry out pilot commands, and advise the pilot as to aircraft and system status, mission progress, threats and alerts. People and agents are part of a socio-technical system, so the functions of the AI and the pilot in our example go well beyond the mechanics of maneuvering the aircraft. Rather, people and agents are inherently part of a network of operations that may involve ambiguous communications and dynamic roles and responsibilities. Combat sorties, for example, require considerable interaction within and across aircraft and remotely with people (and agents) on the ground.

ITSs employ domain models that simulate the behaviors and performance parameters of a diverse range of entities, from space shuttles to tactical radios, including the simulation of human activity – from air battle managers (Freeman, et al., 2003) to village tribal elders (Johnson & Wu, 2008). However, there is seldom evidence of socio-technical factors in conventional approaches to domain modeling. Most modeling frameworks are not adequate to represent the effects of socio-technical phenomena because they focus on functions and tasks rather than group interactions and chronological activities, do not relate perception to activities and physical interaction, and/or focus on prescribed processes rather than practices.

The need for domain models to capture and reflect socio-technical processes arises in most contemporary tutoring applications, as today's workforce must be trained to be proficient in an information-rich, networked, digital, automated world. In some instances, the socio-technical context is not just backdrop enrichment but an explicit focus of the learning; ITSs are being developed that include training for detecting and overcoming problems with socio-technical lapses.

For ITS developers serving such communities, the need to train personnel to overcome such disruptions calls for new domain modeling constructs. Simulation approaches grounded in discrete models sensing and acting independently serve only to exercise how each model behaves and reacts within the confines of the assumptions of the modeler, which are often limited to technical factors. While a communications disruption, for example, would adversely affect discrete entities comprising an overall domain model, it may also have devastating systemic effects on the way people and intelligent systems work together. The unanticipated, subtle socio-psychological effects of factors like denied communications can disrupt coordination and degrade trust. Many unknowns surround how even slight disturbances in team interactions can portend serious downstream consequences.

Domain models are needed that can properly capture work practices of the socio-technical system (Clancey 1993). With teams in complex, highly engineered environments relying increasingly on advanced collaborative and technological work systems, an urgent need has surfaced for a more robust capability to train these teams to succeed when faced by both nominal and adverse conditions. In the rest of this chapter, we present an innovative application of Brahms, a sophisticated and proven work practice analysis simulation (Clancey, et al., 1998; Clancey, et al., 2013). We introduce models and a testbed built to support training, research and concept development for human-automation teams to achieve success in overcoming socio-technical disruptions. Before we present our socio-technical domain modeling approach, we provide in the next section a brief discussion of Brahms to contextualize the remainder of this chapter.

The Brahms Work Systems Framework as Domain Model

Brahms is a work practice modeling and simulation framework used in designing and implementing work systems. It is particularly useful for activity systems with distributed teams with different roles and responsibilities using automated information processing and model-based tools. In that sense, it is well suited for modeling a diversity of domains with rich levels of interpersonal and physical interactivity, as is required for many ITS applications.

Socio-Technical Work Practice Simulation

A work practice simulation represents chronological, located behaviors of people and automated systems. In contrast with task models, which represent abstractly what behaviors accomplish (i.e., technical functions; Schön, 1987), a behavioral model represents what people and systems do, called activities (Leont'ev, 1979; Clancey, 2002). Activities include monitoring (looking, attending), moving, communicating, reading and writing, all of which require time and occur in particular places with other people,

tools, materials, documents, and so on (Suchman, 1987; Lave, 1988; Ehn, 1989; Wynn, 1991). In terms of work, a function/task model abstracts what a person or system does (e.g., "determine location"), while a cognitive-behavioral model of practice represents how the work is carried out in the world (e.g., simulate a person moving, changing the state of a control, perceiving a display's representation, and recognizing a problem). An activity framework enables modeling how knowledge and expertise of distributed teams are applied in practice, which includes noticing and characterizing a situation as being problematic and defining goals and methods for handling the situation (Jordan, 1992; Clancey, et al., 2005).

In simple terms, most cognitive models describe knowledge and operations for transforming information and world states. A Brahms activity model includes this cognitive aspect but casts it within the larger simulation of processes and physical interactions in which operations occur in the world. From a psychological perspective, activities are how people conceive their day-to-day affairs as social actors manipulating tools, representations, and other objects while they move and communicate in some physical setting (e.g., going to the grocery store to buy dinner). Activities are ongoing conceptions of "what I'm doing now" (Clancey, 1997); they encapsulate roles ("whom I'm being now"; Wenger, 1998), norms ("what I should be doing now"; Lave and Wenger, 1991), and progress appraisals ("how well I'm doing"; Feltovich, et al., 2008). In this respect "the context" and "the situation" are conceptual; the Brahms engine is designed to simulate how activity conceptualizations are activated, prioritized, interrupted, and resumed psychologically, as perception, inference, and communications modify the agent's concept of "what I should be doing now."

The Brahms multiagent framework can thus model: (1) people's beliefs and behaviors (their activities, including communicating and moving), (2) objects with behaviors (e.g., instruments, devices, vehicles, computer agents), and (3) representations (e.g., documents, displays, diagrams) (4) in a physical setting (e.g., cars, offices, roads, regions). These are all constructs in the Brahms modeling language, which is a formal programming language that is compiled and interpreted by a Brahms virtual machine (VM). The VM loads compiled models, runs a simulation by interpreting the compiled code, and generates log files and history files that can be used to analyze the results of the simulation. The VM enables Brahms to be integrated with simulations running on other platforms, which may require integration modules using network application programming interfaces (APIs) to establish communications and synchronization.

Although our focus here is domain modeling for ITSs, we note briefly that Brahms simulations can serve many purposes. Brahms was developed as a tool for facilitating systems-level thinking about both formal procedures and implicit cultural practices. As a behavioral simulation, it enables what-if analysis and experimentation with more scenarios having complicated interactions than anyone could either generate manually or anticipate (e.g., see Clancey, et al., 2013). Brahms can serve as a "total system" design tool to promote multidisciplinary collaboration at design time, enabling relating and objectifying perspectives so people are learning about each other's work, methods, and concerns and thus are better able to discover and reconcile design tradeoffs. A Brahms simulation is also useful for communicating designs to other stakeholders, such as managers and adopters. Brahms simulations can generate metrics for predicting probabilities of different kinds of outcomes, costs, timings, etc., to justify further concept elaboration and verify designs; these metrics are more precise than task and workflow models by virtue of representing spatial-temporal interactions in more detail.

Brahms Design Process Overview

Brahms agents can run on different computer platforms and communicate using a variety of network protocols using a service-oriented architecture with VMs. Agents send messages in structured natural language—facilitating interaction with people. These agents can be integrated with simulation-based ITS environments (e.g., to simulate other entities in a tactical simulation), as well as software and hardware

elements. Bringing together the systems engineering phases that Brahms facilitates—analysis, design, evaluation, and experimentation—with implementation itself, Brahms enables a simulation-to-implementation research and development (R&D) methodology (Clancey et al. 2008).

The work systems design approach seeks to properly relate people, technology, facilities, and work processes into a coherent system of interactions. In considering the larger sociotechnical system in which human-automation interactions occur, work systems design aims to develop human-centered systems by which technology is designed as a tool that fits organizational roles, protocols/procedures, communication practices, facilities (e.g., physical layout), and especially the need for strategy, improvisation, and workarounds. Accordingly, we have developed a methodology of empirical requirements analysis that iteratively improves system designs by experiments with prototypes that combine ethnographic observation, work practice modeling, and what-if redesign of the system (Clancey et al. 2005).

To evaluate a work system design and its technology, we need to know how the total system will perform. For example, the effectiveness of an Apache pilot is evaluated in the context of the overall mission objectives. The information it provides and its actions are potentially relevant and important in the coordination and outcome of operations potentially involving many distributed agencies and teams. Work practice simulations enable us to understand and predict how distributed people and automated systems will interact, producing sequences of events that may be unanticipated and together constitute the output or accomplishment of the total system.

Crucially, Brahms models are crafted to be general and adaptable, so they can be configured to simulate a large space of scenarios. This enables verifying that the total system design satisfies success criteria (or fails appropriately) in a wide range of conditions and finding ways of optimizing it under different workload and other contextual assumptions. In this respect, the simulation becomes a research and experimentation tool for defining, exploring, and understanding the range of scenarios that may occur, refining roles, protocols, tools, and automated systems to cope with extreme, non-routine variations that may occur (e.g., see Clancey et al., 2013).

To accomplish this flexibility, people, technology (manned aircraft, unmanned aerial vehicles, computer systems), and the environment (e.g., remote facilities and the topographic model of the operational environment) are modeled as separate entities with their own internal behaviors (i.e., interactive, dynamic processes). Different entities interact by communicating (e.g., data transmission, email, loudspeaker), observing through sensory systems (e.g., human perception), and direct manipulation (e.g., external controls).

In summary, the main idea of work systems design is to develop technology by a systematic, scientific methodology that contextually relates technology to human practices that involve people playing different roles using automated systems in a simulated environment (Sachs, 1995). The work system design methodology incorporates participatory design (making "users" of technology part of the R&D team; Greenbaum & Kyng, 1991), participatory observation (i.e., providing researchers access to the operations team & work setting; Spradley, 1980), ethnographic studies (placing work systems in a socio-cultural context; e.g., Suchman, 1987; Jordan, 1992), and work practice simulation (modeling the behavior and interactions of people and technology on multiple design levels in different simulated conditions). Brahms simulations incorporate a great deal of detail, but provide a way of organizing and understanding the details of different system components and how they interact in time and space.

Domain Modeling: Socio-Technical Factors and Requirements

Domain Modeling Characteristics and Requirements

Developing tutoring systems to train personnel to succeed with each other and in concert with information systems requires domain models that capture the context and contribution of human and autonomous system behaviors, particular for learning objectives related to conditions of degraded command, control, and communications (C3) and automation support. An overall simulation or ITS requires fully interoperable models and valid environments grounded in theories of human cognitive and social systems. The domain modeling must therefore support total system simulations—incorporating models that define and relate assumptions about people and their work practices, roles, and protocols; representational tools and interfaces; facilities and layout; autonomous vehicles and other forms of autonomy; external threats; and the physical environment. This approach to domain modeling can best support training for the complexity and range of possible scenarios in complex, fluid socio-technical environments.

Domain Modeling Gap

Tutoring systems and simulations that situate instruction in socio-technical contexts generally adopt an object-oriented domain modeling paradigm where affordances in a domain (e.g., a computer display) encapsulate a set of behaviors. Behaviors can be scheduled or triggered through an action or event, or by proximity of the user or another entity. While tractable and reusable, this approach is not intended to capture a continuum of disruptions and thus generally embodies fixed assumptions, including conditions (nominal), user roles (immutable), order and duration of tasks (consistent), and access to and reliability of information (uninterrupted and correct).

Another approach that could support more robust socio-technical domain models is workflow modeling, supported by recent industry standards like the Business Process Execution Language (BPEL), an extensible markup language (XML)-based language with structured programming concepts. Workflow models, though, are based on a functional flow-based abstraction that models defined tasks and operations such as those formalized in business procedures and are thus ill suited to capturing the dynamics and uncertainties of complex operations in contested environments. Some systems such as I-X (Wickler, 2007) have an "activity" construct, but the interpretation is in terms of functions (tasks) and plans. Such models do not include or model how "off task" activities are performed, such as managing disruptions and providing assistance to teammates. For this reason they typically do not model "a day in the life" of an individual working with other humans and with automation, particularly under off-nominal conditions.

A different approach to modeling agency in a domain is to apply a Belief, Desire, Intention (BDI) framework (Bordini, et al., 2005). While this approach offers some utility in agent-based modeling, BDI systems are not sufficient because they do not, generally speaking, model objects, people, and systems as independent processes interacting in a simulated environment. For ITS domain modeling, in particular, BDI frameworks would have deficiencies in modeling the "total picture" including the environment (places, buildings, roads), where the agent is located in the world, how perception is affected by the agent's location, how perception is affected by the agent's conception of the present activity, and behavioral interactions among objects and systems. For domain modeling of the type we are discussing, namely, training personnel for human/autonomous operations in denied or contested environments, BDI frameworks are not equipped to model how, when, and where people interact with computer interfaces; protocols of how people carry on a conversation; or chronological activities in "a day in the life" of each agent.

Another, related approach is to employ cognitive modeling principles for capturing the behavior of intelligent agents and systems in a domain model. Commonly used architectures such as Soar and Adaptive Control of Thought—Rational (ACT-R) share with Brahms the basic cognitive distinction between declarative memory (beliefs) and procedural memory (activities/workframes). Cognitive modeling approaches, though, emphasize how behavior is the product of internal neuropsychological encoding, retrieval, latencies, and other cognitive phenomena. While they may model relevant reactive and inferential capabilities, cognitive agent architectures based on theories of human thought and reasoning are not designed to model the sort of embodied theories of attention, deliberation, and action that locate all actions, including mental operations, within a physical setting. This limits the reach of strictly cognitive approaches in modeling how distributed, multi-actor activities, functions, and tasks are accomplished in practice (Lave, 1988).

Bridging the Domain Modeling Gap with Brahms

Domain models for ITSs must support, as we have discussed, a broader socio-technical context, particular for training learners to successfully continue to perform with each other and with their automation systems under degraded circumstances. The gaps presented above can be bridged through a modeling approach using Brahms, because as described above, it is based on socio-cognitive theories of perception, inference, communication, and collaboration (for example, Ehn 1989; Greenbaum & Kyng 1991; Sachs 1995).

Brahms includes the reactive and inference modeling capabilities of other computational architectures based on theories of human thought and reasoning, but is based on an embodied theory of attention, deliberation, and action. Brahms combines an agent's internal state (possible/incomplete activities, plus beliefs, which can represent plans and goals) with a complex modeled environment to determine next behaviors.

In contrast to workflow models, Brahms' activity-based approach represents how functions are carried out in practice (Greenbaum & Kyng 1991; Sachs 1995). This emphasis on "what actually happens in practice" distinguishes Brahms from frameworks that abstract work into functions and tasks. Brahms is a descriptive (behavioral-cognitive) model with prescriptive design implications. In contrast, many representations used for instruction are purely prescriptive/normative; they specify what is supposed to happen, omitting particulars of when, where, with whom, using what tools, etc., that an activity model represents. For example, a workflow model might indicate that a job is conveyed from one office to another; an activity model would indicate how that happens, such as using an online dropbox, using the server for the dropbox, defining who uses the dropbox software and who uses what computer, noting its location, determining how the software is used (e.g., referring to a sheet of paper?), and detailing what the person does when required data are missing. This inclusion of "practical" and "logistic" aspects of work, beyond often well-articulated definitions, rules, and procedures, gives Brahms-style activity simulations advantages for tutoring how to carry out a technical task in a complicated environment.

Brahms is distinguished from BDI systems (e.g., Jason and AgentsSpeak) by modeling an agent's conceptualization of activities as parallel-hierarchical processes in the subsumption architecture. This allows modeling how activities are like identities that blend and contextually change what is perceived, how communications are interpreted, and how tasks are prioritized.

Brahms also contrasts with cognitive modeling frameworks that focus on simulating "the mind of the agent". Of particular importance to modeling human/autonomous operations in austere or contested environments is locating all mental activity and physical actions in space, which Brahms inherently supports. The Brahms engine simulates where every agent and object is located at every clock tick, as well as what

can be heard or seen at each location, such as whether an agent can hear someone speaking nearby. Although not intended to be a general spatial representation language, the Brahms language provides constructs and built-in operations that support modeling the environment and processes within it. If, for instance, an agent/object contains or holds another agent/object, as in an aircraft with a pilot, then the location of the contained object (the pilot) is updated automatically when the object that holds it (the aircraft) moves.

Modeling domains characterized by interactions and, potentially, disruptions to those interactions is support by Brahms' capabilities to simulate chronological, located behaviors, meaning, how functions and tasks are accomplished in practice. Actions involve perception and motion in space and may involve interruptions, metacognition, informal assistance from others, or workarounds. Brahms explicitly models context-sensitive perception, and its subsumption architecture allows composed activities at different levels of abstraction to "run" in parallel (e.g., handling an incoming request is contextual, reflecting the agents' current roles and activity norms). This subsumption architecture distinguishes Brahms from conventional object-oriented programming as workframes may become active and interrupted or resumed in an activity hierarchy.

Brahms is thus designed to simulate the interactive behavior among people, systems, and the environment (e.g., how a pilot interacts with an onboard associate and coordinates with unmanned aerial system [UAS] crew operating supporting aircraft). Such behaviors can be observed, described, formalized, quantified, and predicted in Brahms. Emphasis is on modeling the "total system" to understand and simulate emergent outcomes, which are the product of what is happening in the world.

Brahms-CAST: Socio-Technical Domain Modeling of Denied Environments

In this section we present an example domain modeling effort intended to enrich simulation-based training in a specific tactical context. Although the training system in this example is probably better characterized as a simulation than as an ITS, we believe that the domain modeling aspect of this example illustrates how a socio-technical modeling framework can support technology-mediated training in general, and that the lessons from employing this approach in tactical simulations apply equally well to ITSs.

The abbreviated description of our preliminary design in this section illustrates a socio-technical approach to address the gaps detailed earlier in this chapter. Specifically, we are creating a simulation testbed that models a complete scenario, incorporating the agents, aircraft, and systems for C3, navigation and targeting as well as facilities, geography and threats. This domain model, called the Brahms Contested Airspace Simulation Testbed (Brahms-CAST), can be integrated with simulation-based training environments and enrich learning by incorporating socio-technical factors and effects into a mission scenario (such as in Farkin, et al., 2004). Brahms-CAST can also support the development of ITS, by providing domain models with the ability to exhibit and respond to socio-technical actions and events.

The example focuses on training personnel who work closely with automation systems to detect, diagnose, and work around disruptions. "Disruptions" can interfere with human-human teamwork as well as with interaction between an individual and an automation system. Consider, for example, disruptions to communication. Benign events like atmospheric or solar storms can degrade communications; hostile acts such as jamming, spoofing, and cyber-attacks can also adversely influence human-automation teams. Individuals and teams must be prepared to detect, diagnose, repair, and restore services while maintaining continuity and minimizing risk. And they must do so in close collaboration with their automated systems (which may or may not be reliable or even operational).

Willful disruptions to communication and networks is one of a cluster of hostile actions categorized as anti-access/area denial (A2/AD), a growing concern that is occasioning discussions among military leaders about training. A2/AD is a broad label intended to include threats to US forces' regional access and freedom of action. Because A2/AD strategies may include cyber-attack, electronic jamming, or kinetic damage to network facilities, military personnel must be trained to overcome disruptions to C3 and lapses in automated decision support capabilities while continuing the mission. There is thus a need to train combat forces to overcome threats that alter the way people work with each other and with their information systems.

Scenario

A key requirement in developing the domain model is identifying the critical phase(s) of an overall activity during which emergent interactions may occur and have a bearing on the intended purpose of the simulation, such as planning, training, or developing procedures. Our methodology thus begins with developing a scenario to identify the players and properly contextualize the activities to be modeled. In the case of Brahms-CAST, the scenario is a hypothetical sequence of events depicting an A2/AD action against US air combat forces by a fictitious near-peer adversary. In the nominal, or baseline scenario, four F-16C tactical fighter jets launch to conduct a mission with a standard sequence of preflight, taxi, takeoff, tanking, push point, initial point, target, and return to base. The formation departs Hill Air Force Base (AFB), performs air refueling with a KC-135 Stratotanker, conducts tactical mission checks with the E-3 airborne warning and control system (AWACS) aircraft, and flies to the target in the Tonopah Test Ranges. The scenario defines the players, locations, timelines, and in general terms, the sequence of events.

The top-level entities we created are a formation of four F-16s, the AWACS, and the Stratotanker. The latter two were modeled to fly appropriate flight paths and to communicate with the F-16s for air battle management and air refueling, respectively. We placed greater emphasis on the fidelity of the F-16 models. The adversary agents in the scenario represent the source of electronic warfare (EW) effects (i.e., jamming and spoofing).

We also identified the equipment and systems needed to complete the mission. This enables the Brahms-CAST system to simulate, for instance, the processes activated when one pilot communicates with another. Figure 1 demonstrates the intermediate steps of such a transmission.

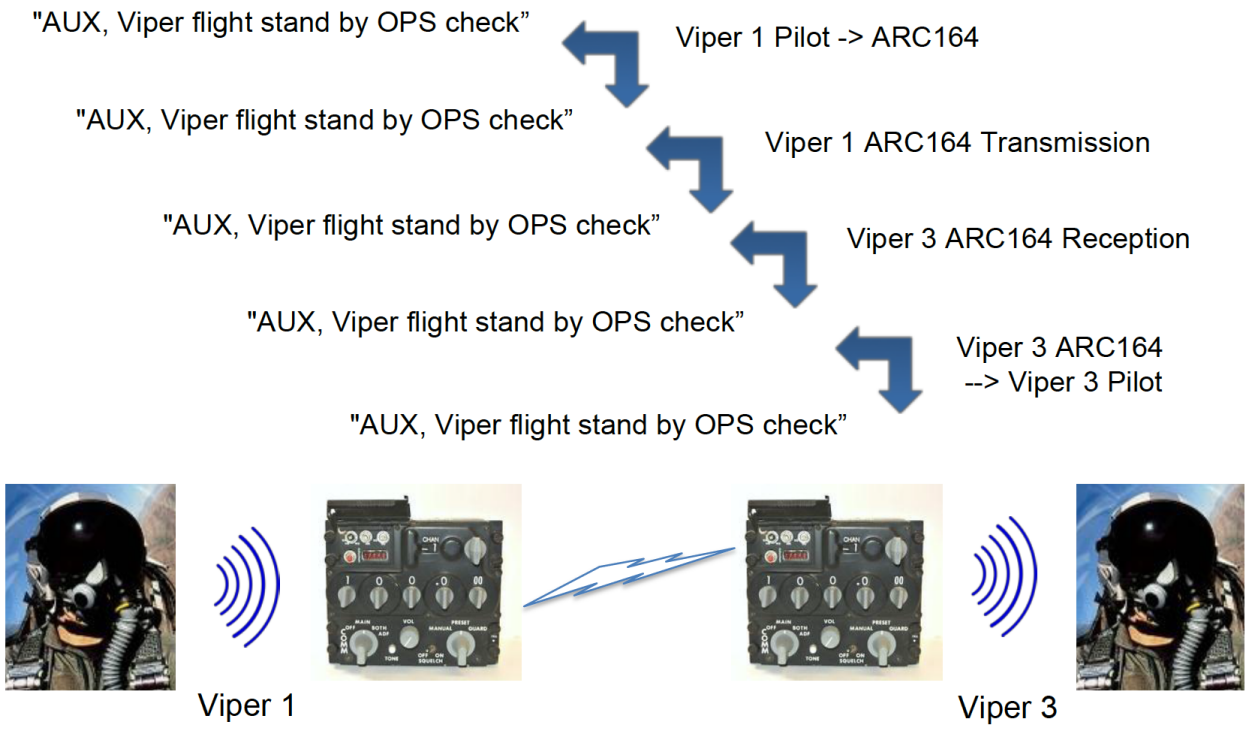

Figure 1. Schematic depiction of socio-technical modeling of air-to-air communication.

Domain Modeling

Once the scenario has been specified, our domain modeling process proceeds in two principal steps: specify (the outcome of which we refer to as a model sketch) and design (the outcome of which is the domain model created in Brahms). While scenarios guide the model specification process, the focus must be on modeling the domain in a general way, including the people, instruments, automated systems, aircraft, and other objects. Conventional approaches to modeling failures and other off-nominal states tend to focus on the details of the particular failure mode but seldom go farther to understand the normal operations in the manner that a general model requires (i.e., a model that can be run on a wide variety of scenarios). These approaches are unlikely to yield models that represent how an activity is carried out normally in practice or how an automated system such as an instrument or display normally operates. A baseline domain model should therefore capture in considerable detail the normal practice for activities that will be affected by scenarios in which special events of interest occur. The specification process is thus performed in great detail because of the importance of getting the models correct – all subsequent scenario runs, variations, and analyses are based upon the correctness and completeness of the model sketch.

From the scenario, we developed the domain model using our work practice methodology and implemented in Brahms. It is important to note that the model, although derived from the scenario, is not a "hard wired" model but a domain blueprint that is variable depending on initial conditions and assumptions. Part of the novel utility of this approach is the ability to run the model over numerous scenarios and analyze outcomes and how different assumptions and conditions can lead to different patterns of results.

The model sketch elaborates the scenario by describing in detail the specific actions, behaviors, and alternative courses of action the agents in the scenario are capable of exhibiting. As a brief example, we discuss the portion of the model sketch devoted to the Push phase of the scenario. As the formation (designated Viper 01 Flight) approaches the push point, they check in with the Airborne Warning and Control

System (AWACS), call sign “Darkstar”. The model sketches both communications and activity, as illustrated in the short excerpt below:

Tactical Check-in:

- Flight Lead checks in with AWACS and passes the following items at a minimum:
    - Flight/Package status with any alibis, direction, requests (as fragged, rolex, msn change, etc.). AWACS will initiate a Bullseye Check and Flight Leads will acknowledge, #2/3/4 by exception. Request Parrot/India check to ensure all Modes/Codes are properly identified.

Aircraft Check-In:

- Bullseye Check: Accomplished during aircraft check-in with TAC C2 (AWACS, call sign “Darkstar”).
    - Viper 01: “AUX, Viper flight push PRI FMT 001, AUX SECURE.”
    - Viper 01: “Viper 01 check FMT 001.”
    - Viper 02-04: “2,3,4.”
    - Viper 01: Viper 01 check AUX SECURE.”
    - Viper 02-04: “2,3,4.”
    - Viper 01: “Darkstar, Viper 01, Checking in as FRAGGED (all A/C are here), request bullseye check.”
    - Darkstar: “Viper 01, Darkstar, bullseye check 360/27, Angels 21.”
    - Viper 01: “Viper 01 Same.”
    - Viper 01: “Darkstar, Viper 01, has AO Update Delta, standing by for additional words.”
    - Darkstar: “Viper 01, Darkstar, AO Update Delta is current.”

Roll call is initiated by AWACS after general communications and tactical parameters have been passed (e.g., words changes, package status, rolex, weather plan, key mission enablers, and LOWDOWN). The roll call order should be briefed during mission planning. Command and control (C2) will poll any flights that do not answer the roll call.

Viper 01 establishes communications (comms) with AWACS and receives all the required updates prior to the push. Viper 01 then enters a holding pattern east of the push point (approximately 135 nautical miles [nm] from the target area). At this time, Viper 01 continues to remain in the holding pattern until a push time of 0400L.

Off-Nominal Variations

The model sketch also defines how failures or disruptions are represented and how the agents respond. For instance, under A2AD-like conditions, upon nearing the target area the pilots begin noticing some minor comms jamming. Specifically some of the transmissions (approximately 25%) are clipped on the Have Quick radio when jamming is active (we use "x" to replace characters in a transmission to denote garbled communications):

- Viper 01: " AUX, Viper 01, flow 18x, Vixxer, x1, set 40x."

In this situation, each pilot in the formation would recognize that an ARC-164 secure radio is being jammed and notify Viper 01. This has a minor effect initially, but as they continue to get closer to the target area the jamming would become progressively worse:

- Viper 03: "AUX, Xxper 01, Vixxx 03, Chaxxermark AUX FMT 001."
- Viper 01: AXX, Vipxx X1 fligxx, Chattermark AUX XMT 001."
- Viper 01: "AUX, Viper 01 flight check HQ FMT 001."
- Viper 02-04 "Viper 02 loud and clear, Viper 03 loud and clear, Viper 04 loud and clear."

At first, the Have Quick radio hopping frequency radios would help but as the 4-ship continues to get closer the four pilots would begin to see degradation in these radios as well. The flight lead might attempt to try other filtered multitone (FMT) channels but with no luck. Furthermore, Viper 01 attempts to contact Darkstar on satellite comms (SATCOM), but SATCOM is jammed as well:

- Viper 01: XXX, Vixxx X1 Flixxx, Cxxxxxxxxxk, XXX FXX 0XX."
- Viper 03: XXX, Vxxer X1, Xxxxr 03, say, axxxn Lxxxx transmxxxxxx?"

Approximately 50 nm from the target area all radios are unusable due to heavy jamming. The F-16s anticipate the comms jamming and continue to press toward the target relying on other equipment other than comms to keep situational awareness. The flight continues to maintain its current formation position using the comms out plan that was discussed in the mission briefing.

The model sketch further defines how conditions influence mission outcomes. Under denial measures such as radio jamming or global positioning system (GPS) spoofing (or both), effectiveness degrades, workload increases, and the reliability of targeting systems erodes. For a strongly degraded scenario, the mission is unlikely to succeed as exemplified in the model sketch excerpt below:

- With GPS spoofing present, the embedded GPS inertial navigation system (EGI) over time begins to think it is in a different geographical position than it really is. To further complicate the problem, approximately 30 nm from the target, Link 16 surveillance and precise participant location and identification (PPLI) tracks begin to fall off line. This is recognized by the wingman and they maneuver their aircraft closer to their flight leads. This action though makes it much more difficult for the wingman pilots to work their radar and targeting pod (TGP) sensors.
- Approximately 25 nm from the target, the F-16s begin searching the target area. Unfortunately, the TGP picture does not display the location that the pilots were briefed to search. The target area is in and around the location of a town/village. The pilots recognize the town, but, looking out

into the dark desert, cannot figure out where the TGP is looking. To further complicate the problem, the wingmen have very limited cross-check time to figure out where their TGP is looking because they are spending so much time trying to stay visual. Furthermore, they are unable to communicate this because of the comms jamming that is present.

Scenario Implementation

The goal of the work being reported here was to implement enough of the model sketch in Brahms to validate the application of this approach to modeling denied environments and assess the viability of Brahms working in concert with other software technologies. We created detailed F-16 models (pilot, aircraft, flight controls, navigation, comms), as well as lower fidelity models of AWACS and Stratotanker aircraft. We also constructed smaller models for air control agencies and adversary electronic warfare (EW) agents.

Models were created within either agent or object hierarchies using the Brahms Composer (Figure). The Brahms Composer is a model development environment supporting classes and subclasses with full inheritance. The Brahms Composer includes a graphical editor for creating agent, object, conceptual object, and geography models, with access to source code editors. The Composer provides a common interface to build, execute, and analyze models. Models created using text editors can be imported. The object class selected in the right pane of Figure 2, for instance, shows the ARC-164 as a subclass of AircraftRadio, inheriting its parent properties (as well as the properties of the classes to which AircraftRadio belongs).

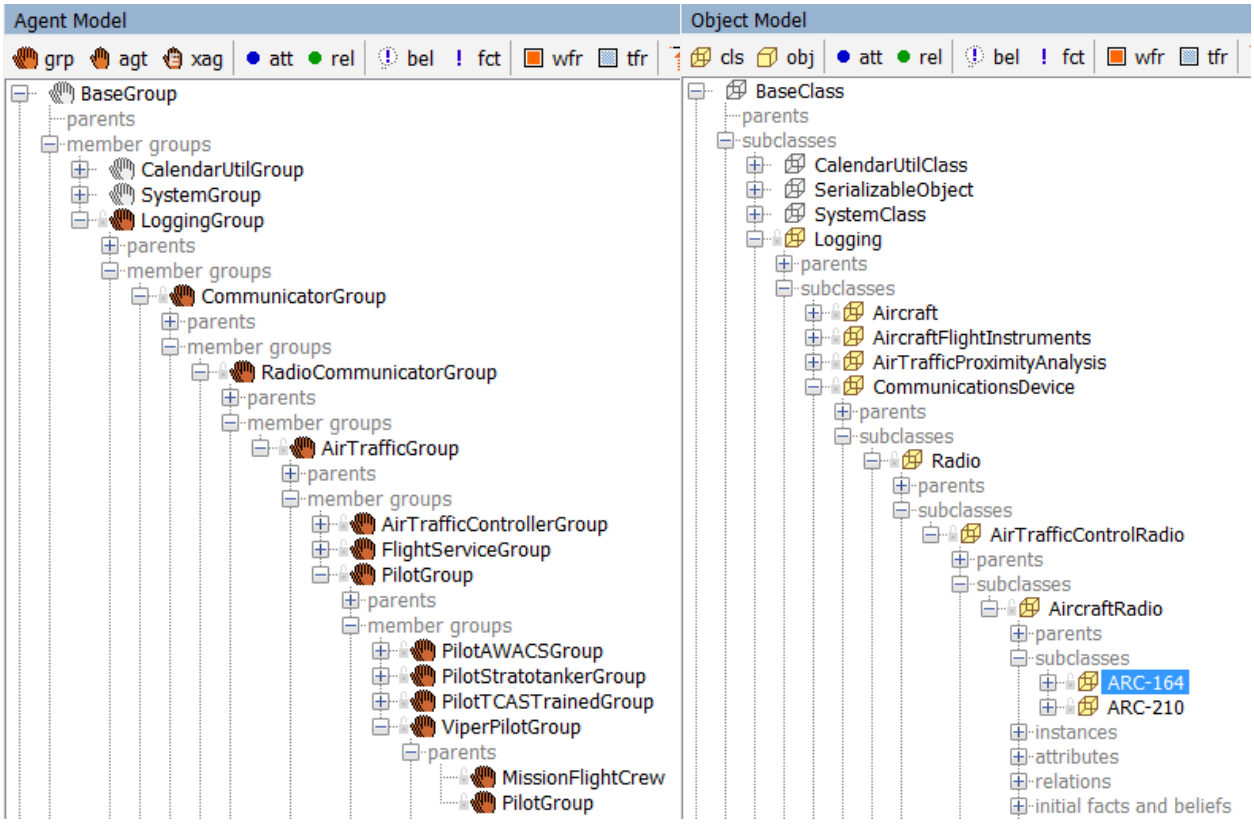

Figure 2. Models in Brahms-CAST showing agent (left) and object (right) hierarchies.

Brahms agents reason about beliefs, desires, and intentions using Thoughtframes. Behaviors in Brahms agents are encapsulated in representations called activities (describing what agents do) and workframes (describing when activities are performed). Figure 3 shows an excerpt from the F-16 pilot model that graphically depicts a portion of the activities hierarchy. Orange boxes are workframes, a torch icon signals activities like communications and moves, and crossed torches indicate composite activities that have several workframes and/or activities within them. For instance, the selected activity in the left pane, "Next_Flight_Route", is part of the composite activity "CheckPlaneLocation".

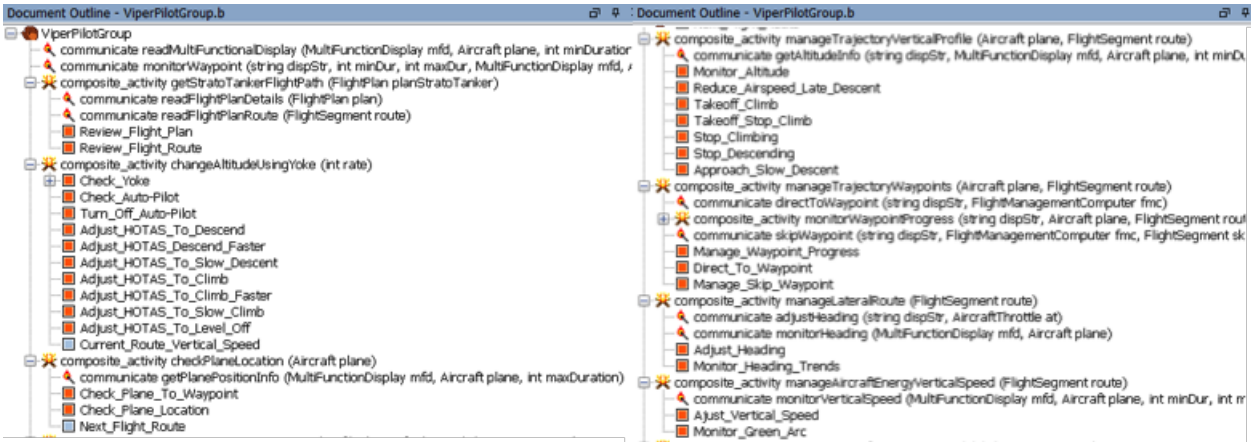

Figure 3. Model excerpt showing F-16 pilot activity hierarchy.

Once models are created, they are run and all event data are published both to an structure query language (SQL) database and to a log file. Multiple runs under differing sets of assumptions permit detailed analyses of how initial conditions and assumptions affect agent performance, mission outcomes, or other measures of interest.

Analyses of scenario runs are supported in Brahms by the AgentViewer, which parses history files generated by the Brahms VM. The AgentViewer provides a detailed trace of all events during a simulation, enabling a modeler to analyze and review the behavior of the agents, objects, and their interactions. The AgentViewer displays a timeline with agent activities, including inferences, communications, and movements. Drilling down displays details of an agent's belief state and world facts at various points in time, linking to workframes and thoughtframes that changed facts and beliefs. It thus permits the analyst to view in numerous ways, time scales, and levels of granularity the activities executed during a scenario run.

We ran scenarios for both and off-nominal conditions. Outputs were exported via a keyhole markup language (KML) export module developed for this project and to the SQL database. A glimpse of the utility of Brahms in simulating important work practice details is offered in Figure 4, which depicts air-to-ground transmissions between the flight lead and departure control early in the mission (vertical blue lines indicate communication). In the context of an ITS, for example, a learner could employ the AgentViewer to drill down to understand how communications are managed and other resources enlisted when workload exceeds timely response to all matters requiring attention, and what the consequences are of various task management alternatives.

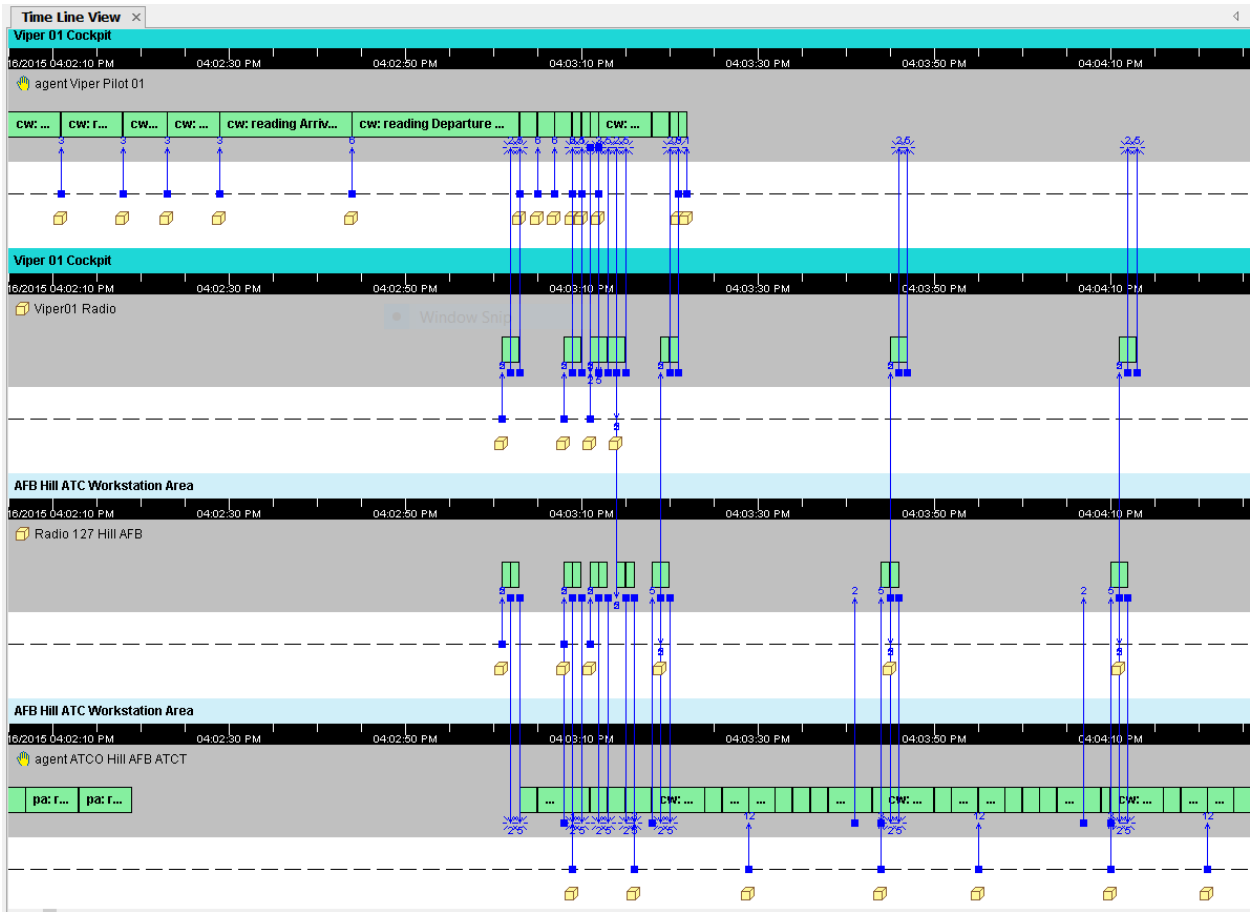

Figure 4. Time-expanded AgentViewer display showing flight lead communications with departure.

Contested Environment Simulations

As a proof-of-concept illustrating Brahms-CAST applied to mission plan evaluation and related assessments, we designed denied scenarios with A2AD effects, and then ran the model against these scenarios. The outcomes data generated from these scenarios demonstrates how Brahms-CAST can be employed for mission plan evaluation.

To model communications jamming and GPS spoofing, we incorporated into the radio models a jammed state during which the radio transmits and receives garbled messages. The level of signal attenuation is modeled by garbling a percent of the message based on distance of the aircraft from the jammer. Garbled messages are modeled as transformed text strings, where GARBLE tokens are substituted randomly for tokens in the message (e.g., "AUX, Viper GARBLE GARBLE, 112 is 256, 346, 1200"). The jamming percent determines the number of GARBLE tokens relative to all tokens in the message. We further model jamming percent as a normal probability function, so, for example, 75% = 6.75 tokens on average (i.e., actual garble for a message with 9 tokens might be 5 tokens or 8 tokens, it is not simply rounded to 7 tokens). Note that transmissions are not actually text strings but structured objects, which enables richer simulation of the effects of denials and countermeasures on mission outcomes.

We created a test variation matrix to capture which conditions were being varied and how for each scenario. Two variables manipulated in the matrix are GPS spoofing and radio jamming. Radio jamming can

have a value of “off”, “always”, or “probabilistic” (governed by a probability function). GPS spoofing can have a value of “off”, “denied always”, “denied probabilistic”, “spoofed always”, or “spoofed probabilistic”. These variables, arranged in the test variation matrix, yield 15 variants that are incorporated into the test plan.

We developed probability functions that determine the conditions for any given scenario. The jamming/spoofing (J/S) probability function is defined as (D – d)/D, where D is the distance between the push point and target, and d is the aircraft’s distance from the target. In other words, the probability of jamming/spoofing is inversely proportional to the aircraft’s distance from target. This suggests how Brahms-CAST can be used as an evaluation testbed to simulate mission plans and gather metrics on mission outcomes. Note that probability functions can be arbitrarily complex in order to more realistically model the effectiveness of A2AD tactics and countermeasures.

We capture the results in the SQL database and event logs for analysis and use the KML export module to view the results in Google Earth Pro. The behaviors presented below are for demonstration purposes and do not realistically model jamming technologies or actual operations. For instance, in the jamming scenario the wingman is unable to locate the target, though in practice could continue to the target; theater rules of engagement would define lost comms procedures and 4th-generation fighters would likely remain in visual contact in any case.

For our experimentation under jamming conditions, results show that the flight lead is able to locate the target (Figure 5). However when we investigate the outcomes, we notice that comms were disrupted and that Viper 02 was unable to locate the target. Figure 6 shows three panels from the AgentViewer depicting jamming (left panel), garbled voice communications (center panel), and the resulting failure of Viper 02 to locate the target (right panel).

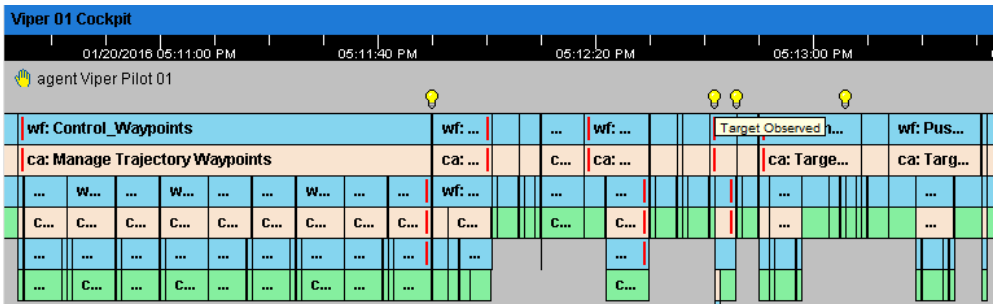

Figure 5. Viper 01 locating target in jamming conditions.

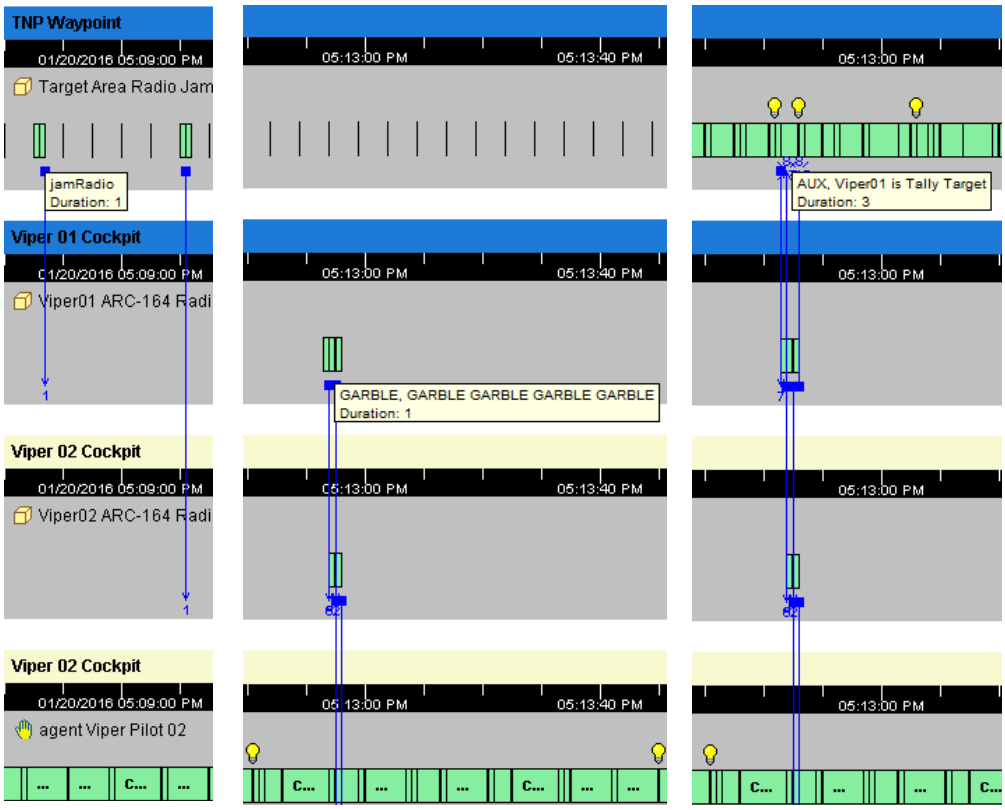

Figure 6. Jamming effects (left) result in garbled comms (center) and Viper 02 failing to locate target (right).

To model GPS jamming effects, we introduced a delay in the GPS receiver that causes a lag in updates to navigation display. This time delay is not intended to be a valid model of how GPS jamming actually manifests but will affect simulations of where the aircraft is headed during piloted flight to a target location. The GPS jamming device is modeled as an object located at the target location. This object monitors for aircraft within 100 nm of its location. The strength of the jamming effect is modeled to be configurable and depends on the distance of the aircraft from jamming device; the closer an aircraft is to device, the more it is subject to jamming.

For GPS spoofing scenario runs, we located a GPS spoofing object (an abstraction of an EW emitter) at the target shelter. In a typical spoofing scenario run, as an aircraft approaches, the spoofing object generates erroneous GPS coordinates and sends these coordinates to the F-16's GPS receiver. The left panel in Figure 7 shows one of the F-16s' radio ("Receiver Viper 01") receiving signals transmitted by this spoofing object model during the mission timeline. The center panel in Figure 7 shows a portion of one of the F-16 pilot models as it approaches the target and erroneously diverts away from the target. The right panel in Figure 7 shows one of the F-16s' TGP object model searching a target location offset from the desired location as a result of the spoofing. To visualize a GPS spoofing scenario, Figure 8 shows the flight lead successfully completing a target pass but the rest of the formation beginning to exhibit the effects of spoofing, deviating from course, and missing the target.

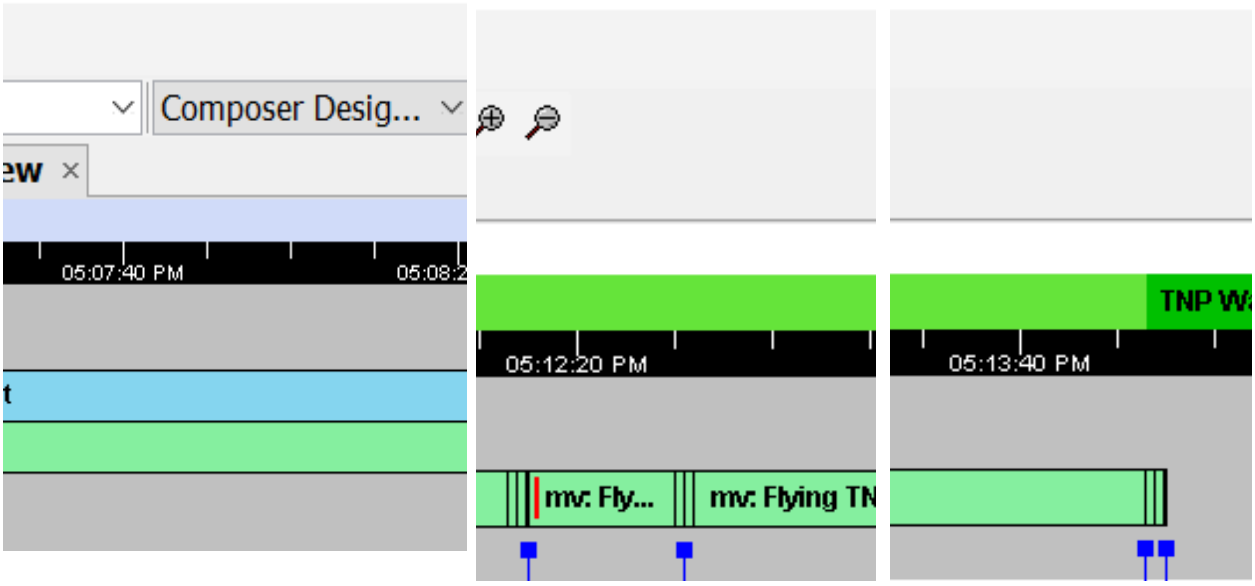

Figure 7. GPS spoofing effects (left) trigger erroneous heading change (center); targeting pod error (right).

Figure 8. Map view of course deviations caused by GPS spoofing, as target is approached and missed.

The utility of Brahms-CAST as a domain modeling tool for analysis and experimentation arises from the ability to run the model numerous times for any given set of initial assumptions. Outcomes can be tabulated as depicted in the example in Figure , where parameters like whether the target was located and how close each aircraft was to the target location can be collected and analyzed. Our tactical results are not intended to be realistic, since assumptions in the model are based on unclassified information and simplifying assumptions. These demonstrations, however, illustrate the potential of Brahms for supporting a diverse array of mission planning evaluations, after-action reviews, contingency analyses, and other assessments.

Run	GPS Spoofing Starts	Target Observed?				Target Observed Time				Target Observed at Distance (nm)				Closest Distance to Target (nm)			
		Viper01	Viper02	Viper03	Viper04	Viper01	Viper02	Viper03	Viper04	Viper01	Viper02	Viper03	Viper04	Viper01	Viper02	Viper03	Viper04
1	1:15:56	Yes	Yes	Yes	No	1:14:30	1:15:17	1:14:17		9.98	7.68	11.91		1.05	0.45	0.44	3.05
2	1:14:13	Yes	Yes	Yes	Yes	1:14:53	1:14:34	1:16:00	1:15:05	10.11	11.29	5.68	9.77	1.1	0.73	0.73	0.46
3	1:13:13	Yes	Yes	Yes	Yes	1:13:13	1:15:23	1:14:19	1:16:24	11.24	4.21	12.34	10.67	0.43	0.73	0.74	1.67
4	1:13:21	Yes	No	Yes	Yes	1:13:31		1:15:21	1:16:01	7.9		10.25	9.8	0.43	2.97	6.25	1.45
5	1:14:24	Yes	Yes	Yes	Yes	1:13:58	1:14:36	1:14:55	1:14:55	7.45	9.88	10.41	12.14	0.42	0.44	0.84	0.71
6	1:12:32	Yes	Yes	Yes	Yes	1:12:13	1:15:00	1:15:50	1:16:07	8.31	10.98	12.44	10.76	6.09	6.26	1.47	1.06
7	1:13:36	Yes	Yes	Yes	Yes	1:14:02	1:13:52	1:14:06	1:14:44	9.25	12.98	12.22	10.64	0.84	0.96	1.61	5.25
8	1:16:40	Yes	Yes	Yes	Yes	1:14:07	1:13:42	1:14:16	1:16:13	11.04	11.42	10.38	7.09	0.7	0.5	0.46	1.48
9	1:11:50	Yes	No	No	Yes	1:13:39			1:15:55	10.2			9.22	0.8	8.99	8.97	1.48
10	1:12:52	Yes	Yes	Yes	Yes	1:13:14	1:14:36	1:15:14	1:15:39	10.08	10.84	9.84	10.96	0.63	6.2	1.46	0.99

Figure 9. Tabulating GPS spoofing results for given initial conditions and assumptions.

Of greatest relevance to ITSs is that, as a generative simulation, Brahms offers a mechanism for a tutoring system to vary assumptions and conditions in the domain model, which can enable an adaptive tutor to modify the level of difficulty or to emphasize key principles.

Discussion: Socio-Technical Factors

This series of events is described above in a manner similar to how conventional modeling paradigms could simulate this scenario. However an ITS or simulation can be far more effective if the underlying domain modeling supports simulation of the socio-technical effects, such as the interaction of ongoing coordinated activity, movement, perception, and manipulation of instruments.

There are numerous interacting factors, not generally accommodated by most modeling approaches, that a socio-technical paradigm could capture and thus enrich a simulation or ITS. A circumstantial combination of events can lead to unanticipated interactions such that people, objects, and systems become causally interdependent with undesirable feedback. Simulating these as independent processes in a simulated world, as in Brahms, has the potential to reveal emergent effects. For example, jamming mentioned above would increase the overall stress level of the pilots (they hear tones in their cockpit from the jamming). Turning down the radio to reduce the volume of the ringing makes it difficult to hear radio transmissions, so communication among the team is degraded. Under conditions of SATCOM jamming, pilots can no longer receive broadcast control from AWACS, which would as a result spend more time checking their air-to-air radar to ensure that no hostile aircraft are airborne. They also devote time during their cross-checks to ensure no surface-to-air missile (SAM) systems are active and emitting. Less time is spent looking at their TGP and at the target area. These user actions and configurations shape the information landscape a user operates in and influence workload, stress, coordination, and team effectiveness. Loss of routinely automated functions requires shifting activities to a different mode of operation that makes accomplishing the mission more difficult.

In particular, when the four ships in the scenario change to a 2+2 line abreast, lost comms means the pilots devote more time to looking outside to confirm the formation is correctly. Under conditions of Link 16 dropout, tightening the formation makes it much more difficult for the wingmen to work their radar and targeting pods. The spoofing and jamming also require each pilot to align the TGP to look in the direction of the target by correcting where the pod is currently looking. This can be a difficult task for the pilot while maintaining a good cross-check with navigation instruments as well as looking outside to maintain formation. This all suggests the need for socio-technical domain modeling that captures the interactions among pilots, aircraft, Link 16, radar and targeting systems in terms of the pilot's overall coordinated activity of monitoring/perceiving, inferring, communicating, and manipulating systems.

Implications of Socio-Technical Representations for ITS Domain Modeling

Domain models generally support a simulation or ITS by furnishing an environment that responds to user actions and generates and correctly propagates events. As our example illustrates, Brahms offers a capability to create a large collection of possible outcomes from initial conditions, each of which documents and reflects the socio-technical factors shaping an event sequence. Brahms-CAST is therefore not merely a hand-crafted, one-off replication of an A2/AD scenario. Rather, it consists of a generalization of roles and automated systems (e.g., pilots, air battle managers, radar, navigation and communication systems, etc.) that play a role in contested environments. Rather than representing only the states and behaviors of subsystems that obtain at a given point in time, Brahms-CAST represents nominal states and behaviors and allows for them to be configured for a diversity of scenarios to characterize alternative behaviors, including absent, alternative, and dysfunctional or off-nominal forms (e.g., dropped communications; inoperative navigation). Initial conditions can be configured to simulate infinitely different workloads (e.g., equipment readiness, threats, weather).

In this way, domain models can be employed to generate scenarios that ITS authors can sequence as part of a syllabus. Each of the many possible configurations of Brahms-CAST parameters (initial facts, beliefs, and properties/states) defines a scenario. The combinations of all possible parameter settings define a space of scenarios that Brahms-CAST is able to simulate with operational validity.

Brahms also enables ITS domain models that have probabilistic outcomes, deriving from (1) variable durations of primitive activities (those not modeled as composite conditional actions) and (2) how perception (e.g., noticing an indicator on a display) is modeled contextually as probabilistic "detectables." Thus, each simulation run of a given scenario (pre-defined initial conditions, e.g., location and effectiveness of the jamming and spoofing sources) produces time-space-state interactions with potentially different outcomes, some with emergent, potentially unanticipated sequences of events that reveal the effects of proximity and timing (e.g., when a pilot scans a display and notices a problem). A behavioral simulation with a short time increment (e.g., 1 s) is particularly advantageous for simulating such effects. Continuing our example scenario, it possible that in some simulation runs of Brahms-CAST a wingman notices the GPS spoofing and advises the lead, far enough in advance of reaching the target area to allow for compensatory measures.

Training developers can use this capability to examine complexities, workloads, and outcomes that result from different sets of initial conditions. Brahms-CAST components can also be adapted to model many related work systems (e.g., different aircraft, mission sets, threats). This flexibility to define additional combinations of people and aircraft of different types enables ITS authors to develop training variations of existing and future systems. Of particular interest in this example are those training systems that address new forms of autonomy and change how people interact with automation under different environmental conditions (e.g., A2/AD).

Brahms-CAST can also help ITS authors develop assessments, by revealing how the timing of events at the level of a few seconds can make a substantial difference in outcomes. In particular, Brahms-CAST simulates how subtle issues of timing in human-automation interactions arise when degraded or missing subsystems result in lack of information and inability to communicate, transforming a given configuration of routine tasks in a normal work system to a situation too complex for the overall work system to successfully manage.

In our example, the events in a tactical sortie reveal how people develop work practices in which they rely on automation, and how the absence of automation may cause the workload to increase and the evolving situations to become too causally codependent to appropriately prioritize tasks or delegate responsibility. That is, the workload has become cognitively complex relative to the person's knowledge, beliefs, roles,

habitual procedures, and tools (see Clancey et al. 2013 for a detailed example of emergent human-automation situations in air traffic control). This framework is thus ideally suited to train personnel in mitigating A2/AD effects.

Brahms-CAST demonstrates the strength of the Brahms framework for simulating behaviors of asynchronous (or loosely coupled), distributed processes in which the sequence of interactions among people and automated systems becomes mutually constrained and unpredictable. Creating and experimenting with Brahms work practice models reveal system interactions that may be omitted, glossed over, or difficult to comprehensively describe in after-action reviews. The simulation can be crafted to generate metrics that can be compared to observational data and/or make predictions for redesign experiments.

Conclusion

In this chapter, we apply a socio-technical modeling framework, called Brahms, that ITS and simulation developers can adapt as a domain modeling approach. Brahms is designed to simulate activities as chronological, located behaviors, that is, how functions and tasks are accomplished in practice. Activities involve perception and motion in space that are modeled explicitly through direct support of the Brahms language and engine (e.g., perception is context-sensitive). In general, activities might involve interruptions, metacognition, informal assistance from others, or workarounds, which can be simulated within the Brahms architecture.

The activity modeling and simulation framework at the heart of Brahms-CAST offers ITS authors a powerful way to think about domain knowledge and expertise because it integrates cognitive processes (perception, inference, belief maintenance) with procedural, interactive behaviors (sequential conditional actions) in a modeled "geography" of areas and paths, an environment with objects/systems having their own behaviors (e.g., navigation displays, unmanned aircraft). Agent (and object) properties and behaviors can be inherited from groups (classes), facilitating model creation and reuse. Through multiple inheritance and blending, behaviors can be represented at different levels to model the effects of training and group practices, as well as individual preference. This framing of work practice in terms of activities, in contrast with task-oriented workflow models, facilitates simulating interacting, choreographed joint behaviors of a team in which individual agents may have complementary, incompatible, and/or the same capabilities.

Modeling domains and expertise in such an activity framework helps training developers capture nuances of the domain that are fundamental to what must be understood and controlled. In this chapter, we discussed an example of how coordination of assets in contested environments becomes highly dynamic and complex when communications are unreliable, intermittent, or not secure. In particular, the constructs provided by Brahms helps ITS creators model how data uncertainty and disengaged command will require humans and autonomous systems to adapt quickly, reconfiguring operational strategy and decision-making authority.

References

Bordini, R. H., Dastani, M., Dix, J. & Seghrouchni, A. E. F. (Eds.) (2005) Multi-Agent Programming: Languages, Platforms and Applications. New York, NY: Springer Science+Business Media, Inc.

Clancey, W. J. (1993) The knowledge level reinterpreted: Modeling socio-technical systems. International Journal of Intelligent Systems, 8(1), 33–49.

Clancey, W. J. (1997) Situated Cognition: On Human Knowledge and Computer Representations. New York: Cambridge University Press.

Clancey, W. J., Sachs, P. Sierhuis, M. & van Hoof, R. (1998) Brahms: Simulating practice for work systems design. International Journal of Human-Computer Studies, 49, 831–865.
Clancey, W.J. (2002) Simulating activities: Relating motives, deliberation, and attentive coordination, Cognitive Systems Research 3(3) 471–499, September, special issue on situated and embodied cognition.
Clancey, W.J, Sierhuis, M., Damer, B., Brodsky, B. (2005) Cognitive modeling of social behaviors. In R. Sun (Ed.), Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation, pp. 151–184. New York: Cambridge University Press.
Clancey, W.J., Sierhuis, M., Alena, R., Berrios, D., Dowding, J., Graham, J.S., Tyree, K.S., Hirsh, R.L., Garry, W.B., Semple, A., Buckingham Shum, S.J., Shadbolt, N. and Rupert, S. (2007) Automating CapCom using Mobile Agents and robotic assistants. NASA Technical Publication 2007–214554.
Clancey, W.J., Sierhuis, M., Seah, C., Buckley, C., Reynolds, F., Hall, T., Scott, M. (2008) Multi-agent simulation to implementation: A practical engineering methodology for designing space flight operations. In A. Artikis, G. O'Hare, K. Stathis & G. Vouros (Eds.), Engineering Societies in the Agents' World VIII. Athens, Greece, October 2007. Lecture Notes in Computer Science Series, Volume 4870. Heidelberg Germany: Springer, pp. 108–123.
Clancey, W.J., Linde, C., Seah, C., Shafto, M. (2013) Work Practice Simulation of Complex Human-Automation Systems in Safety Critical Situations: The Brahms Generalized Überlingen Model. NASA Technical Publication 2013–216508, Washington, D.C.
Ehn, P. (1989) Work-oriented design of computer artifacts. L. Erlbaum Associates Inc., Hillsdale, NJ.
Farkin, B., Damer, B., Gold, S., Rasmussen, D., Neilson, M., Newman, P., Norkus, R., Bertelshems, B., Clancey, W.J., Sierhuis, M., Van Hoof, R. (2004) BrahmsVE: From human-machine systems modeling to 3D virtual environments. Proceedings of the 8th International Workshop on Simulation for European Space Programmes (SESP 2004), Noordwijk Holland, October 19–21.
Feltovich, P. J., Bradshaw, J. M., Clancey, W. J., Johnson, M., Bunch, L. (2008) Progress appraisal as a challenging element of coordination in human and machine joint activity. In A. Artikis, G. O'Hare, K. Stathis & G. Vouros (Eds.), 2008, Engineering Societies in the Agents' World VIII. Lecture Notes in Computer Science Series (pp. 124–141). Heidelberg Germany: Springer.
Freeman, J., Diedrich, F. J., Haimson, C., Diller, D. E., and Roberts, B. (2003). Behavioral representations for training tactical communication skills. In Proceedings of the 12th Conference on Behavior Representation in Modeling and Simulation, Scottsdale, AZ.
Greenbaum, J. & Kyng, M. (Eds.) (1991) Design at Work: Cooperative design of computer systems. Hillsdale, NJ: Lawrence Erlbaum.
Johnson, W.L. and Wu, S.M. (2008). Assessing aptitude for learning with a serious game for foreign language and culture. In B. Woolf, E. Aimeur, R. Nkambou & S. Lajoie (Eds.), International Conf. on Intelligent Tutoring Systems (pp. 520–529). Berlin: Springer-Verlag.
Jordan, B. (1992) Technology and social interaction: Notes on the achievement of authoritative knowledge in complex settings. IRL Technical Report No. IRL92-0027. Palo Alto, CA: Institute for Research on Learning.
Lave, J. (1988) Cognition in practice. Cambridge: Cambridge University Press.
Lave, J. and Wenger, E. (1991) Situated learning: Legitimate peripheral participation. New York: Cambridge University Press.
Leont'ev A. N. (1979) The problem of activity in psychology. In Wertsch, J. V. (editor), The concept of activity in soviet psychology (pp. 37–71). Armonk, NY: M. E. Sharpe.
Sachs, P. (1995) Transforming Work: Collaboration, Learning, and Design. Communications of ACM 38(9), 36–44.
Schön, D. (1987) Educating the reflective practitioner. San Francisco: Jossey-Bass Publishers.
Spradley, J. P. (1980) Participant observation. Fort Worth: Harcourt Brace College Publishers.
Suchman, L. A. (1987) Plans and situated actions: The problem of human-machine communication. Cambridge: Cambridge Press.
Wenger, E. (1998) Communities of practice: Learning, meaning, and identity. New York: Cambridge University Press.
Wickler, G., Tate, A. & Hansberger, J. (2007) Supporting Collaborative Operations within a Coalition Personnel Recovery Center. Paper presented at the International Conference on Integration of Knowledge Intensive Multi-Agent Systems, Waltham, MA.
Wynn, E. (1991) Taking practice seriously. In J. Greenbaum and M. Kyng (Eds.), Design at work: Cooperative design of computer systems (pp. 45–64). Hillsdale, NJ: Lawrence Erlbaum Associates.