Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)

Carbon Copy

The Benefits of Autonomous Cognitive Models of Air Traffic Controllers in Large-Scale Simulations

Steven Estes, Craig A. Bonaceto, Kevin Long, Dr. Scott H. Mills, & Dr. Frank Sogandares
The MITRE Corporation
McLean, VA
sestes@mitre.org

Abstract—**NextGen proposes a suite of new technologies and procedures to be employed by the en route air traffic controller. Typically, the benefits of NextGen concepts are evaluated using fast-time simulation systems. However, these systems often ignore the human component of the National Airspace System (NAS) or represent it with little fidelity. Providing better estimates of NextGen productivity and efficiency benefits requires, in part, better models of human performance. In this paper we describe the construction and use of a cognitive model, Edgar, designed to address this problem.**

Keywords-cognitive modeling; human performance; controller agents; controller worklaod

I. Introduction

Every year at research institutions across the country hundreds, if not thousands, of controllers and pilots participate in human-in-the-loop evaluations. Each controller or pilot typically spends several hours of their free time doing what they do during their work time; flying airplanes or keeping airplanes separated. By participating in these evaluations they help us understand and quantify how, for example, a new Cockpit Display of Traffic Information (CDTI) will affect a pilot's workload or how advanced automation will change the way a controller does his or her job. While these evaluations tell us a lot about the effect of new technologies and procedures on the individual, it is often difficult to extrapolate the inverse effect; how the individual, equipped with these new technologies, will impact the National Airspace System (NAS). And, where fast-time simulation and the like are employed to do so, the human component of the system is represented with little or no fidelity. Ideally, the NAS-wide benefit of any new technology would be tested using huge, human-in-loop evaluations where the human, technology, and environment are all represented with a high level of fidelity.

Of course, even with as many controllers and pilots as are participating in evaluations today, this huge, high fidelity evaluation would require many orders of magnitude more. During these evaluations each controller or pilot might spend a week or more controlling traffic and flying planes in concert with hundreds of other controllers. These evaluations would cover many control centers' worth of sectors and would run, literally, 24 hours a day, 7 days a week. This is, of course, a completely infeasible scenario. If large-scale simulations for the purpose of getting a glimpse of the system-wide picture are to be accomplished, bringing in hundreds of controllers for a single evaluation is not the answer. What is needed is a reasonable controller facsimile; a carbon copy of sorts.

In this paper we will discuss how human behavior can be replicated using cognitive modeling, the application of those models to the aviation domain, and describe our cognitive model, Edgar, and how it can be used in large-scale evaluations.

II. Autonomous Cognitive Modeling

The phrase "autonomous cognitive models" may or may not have any meaning to you, the reader. Those who are unfamiliar with the term might, reasonably, infer something like A.I. (artificial intelligence). And, as a touchstone, this is a fine place to begin thinking about autonomous cognitive models. In the sense that cognitive models are software based agents that act autonomously, there is a definite commonality. There is, however, also a critical distinction between the two.

While some A.I. agents may be designed to seem human, there is no constraint that they must go through a "human-like" decision making process. A cognitive agent, however, attempts to emulate not only observable behavior, but also the process by which the human arrives at that behavior. Further, as will be discussed later in this paper, the cognitive agent has a set of resources that are designed to be human-like. That is, the cognitive agent has human-like resources like memory and vision, all of which are limited in the same way the corresponding human system is limited. Before delving much deeper into what cognitive modeling is and how we have applied it to the air traffic control domain, it will be helpful to provide some context.

A. A Cognitive Science Primer

Prior to the 1950s, cognitive science was a non-entity. This is not to say that studies of human cognition had not occurred until that time; they certainly had. Theory foundational to cognitive science had been generated as far back as Aristotle and Plato, but there had not been a system of study to which that theory had been incorporated and applied. But in the late fifties, the domain of cognitive science began to crystallize, and disparate theories of cognition came together [18]. Critical to

MITRE SPONSORED RESEARCH

the discussion at hand, this coalescing of ideas lead to theories of the mind and accompanying models of attention, memory, language, learning, and the like.

B. The Model Human Processor

In 1983, Card, Moran, and Newell [6] introduced the Model Human Processor (MHP). The MHP was interesting because it not only provided a box-and-arrow model of human cognitive processes, it also acted as an engineering tool, enabling the user to make quantitative predictions about human performance. As shown in figure 1, Card, Moran, and Newell gathered much of the quantitative research over the 30 years prior and applied it across the various constructs of cognitive psychology, describing the capacity, durability, and speed of various components of the cognitive system.

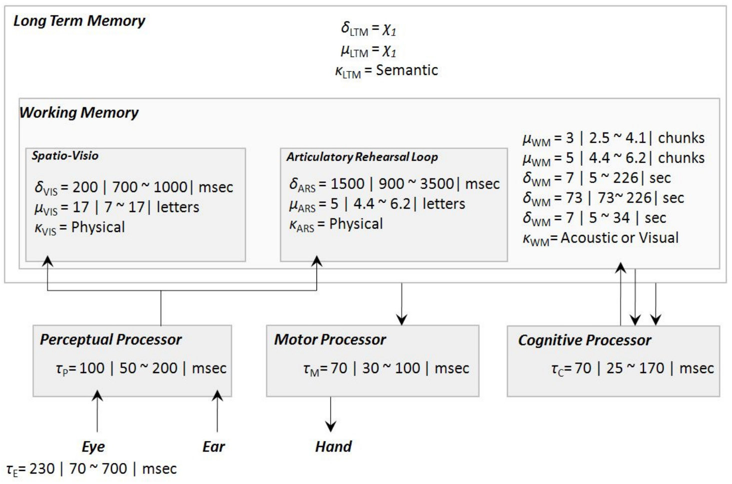

Long Term Memory
$\delta_{LTM} = \chi_1$
$\mu_{LTM} = \chi_1$
κ_{LTM} = Semantic
Working Memory
Spatio-Visio
δ_{VIS} = 200 | 700 ~ 1000 | msec
μ_{VIS} = 17 | 7 ~ 17 | letters
κ_{VIS} = Physical
Articulatory Rehearsal Loop
δ_{ARS} = 1500 | 900 ~ 3500 | msec
μ_{ARS} = 5 | 4.4 ~ 6.2 | letters
κ_{ARS} = Physical
μ_{WM} = 3 | 2.5 ~ 4.1 | chunks
μ_{WM} = 5 | 4.4 ~ 6.2 | chunks
δ_{WM} = 7 | 5 ~ 226 | sec
δ_{WM} = 73 | 73 ~ 226 | sec
δ_{WM} = 7 | 5 ~ 34 | sec
κ_{WM} = Acoustic or Visual
Perceptual Processor
τ_P = 100 | 50 ~ 200 | msec
Motor Processor
τ_M = 70 | 30 ~ 100 | msec
Cognitive Processor
τ_C = 70 | 25 ~ 170 | msec
Eye
Ear
Hand
τ_E = 230 | 70 ~ 700 | msec

Figure 1. Card, Moran, & Newell's Model Human Processor (MHP)

Cognitive models (which may also sometimes include Human Performance modeling) then provide physical, workable representations of the way a human "thinks" and interacts with his environment. Cognitive models may be instantiated as paper and pencil or software models, the primary focus here is on software based models. As such, we can also say that cognitive models are, for the purposes of our research, autonomous agents. Though many cognitive architectures, where an architecture is formal representation of both behavior and processes, are capable of learning [2, 21], the models employed for this project represent only expert behavior and never learn in the course of execution. These models [14] are built by reconstructing the method used to accomplish a given task through controller interviews and literature review. As an example, if the goal is to save an electronic document, the method to accomplish that goal might be annotated as follows:

```
Method_for_Goal: Save_Text File
  Step 1. Look_at Menu_Item whose Type is "File"
  Step 1. Point & Click on the "File" menu
  Step 2. Point & Click "Save As"
  Step 3. Determine File Name
  Step 4. Hands to Keyboard
  Step 5. Type File Name
  Step 6. Hands to Mouse
```

MITRE SPONSORED RESEARCH

This provides a description of the method a person might use to save a file. However, cognitive models go one step further, executing that method in accordance with a set of constraints like those shown in Figure 1. Those constraints will act on the motor, perceptual, and cognitive resources of an architecture in a variety of ways. We broadly define these limits as being either temporal or resource constraints.

A temporal constraint ensures that each step is carried out according to an empirically derived human execution time, despite the fact that a computer could clearly execute any one of the steps in the above example far faster than any human could. For instance, as it takes, on average, a human 1100 ms to point and click on an object on a display, the cognitive agent should likewise also take roughly 1100 ms [6]. Depending on the fidelity of the architecture, this constraint may act at the task or symbolic level and may be bounded or enhanced by rules like Fitt's law [10], which gives a more precise execution time estimate based on the distance to and size of the display target.

Resource constraints are slightly different, in that they determine the availability and/or the capacity of cognitive, motor, and perceptual resources. In its simplest form, this means that a properly constrained cognitive model cannot, for instance, "see" two things at once or "say" two things at once. Depending on the architecture's fidelity, it may also be that working memory resources have capacity and durability limitations in accordance with human cognitive limits so that a model's working memory is not functionally limitless.

So, it is through this representation of human goals and methods, executed according to a set of human-like constraints, that a cognitive model is arrived at that might reasonably replicate human behavior.

III. Cognitive Modeling In The Aviation Domain

Cognitive modeling is a tool and as with any tool, it is most effective when applied according to its expressed purpose. The types of behavior to which this style of cognitive modeling are well suited are characterized, broadly speaking, by experts engaged in routine tasks within a relatively structured (i.e., highly rule based) environment. Typically the task to be completed would involve a human computer interface. By this definition, the task of air traffic control is quite amenable to cognitive modeling.

Of course the tool's appropriateness does not automatically assure utility. Whether or not cognitive modeling may be usefully applied to problems of the aviation domain, the more basic question is, "When does cognitive modeling provide a unique and/or better solution than other analysis tools?" The answer is found in the concept of the Cognitive Model in the Loop Evaluation (CITL).

A. The CITL versus the HITL

As touched on in the introduction, an ideal HITL evaluation is one that might be called a 24 x 7 x 2 x 12 HITL. This is an evaluation that runs 24 hours a day, seven days a week over two weeks and involves twelve or more controllers, each working different sectors. The impracticality of such an evaluation is obvious, as should be its benefits. With this number of controllers available over such a long period of time, evaluations could include much larger geographic areas, and, as an added benefit, would not require the type of contrived, heavily scripted scenarios that are often employed to ensure specific events occur during the evaluation. The CITL concept thus allows on-demand access to dozens of surrogate controllers that can be used in exactly this type of idealized evaluation.

CITLs may also be used in a hybrid HITL in which real controllers control traffic in the center sectors, but bordering sectors are controlled by cognitive models. This would allow air traffic to be delivered in a more realistic manner to the controller and, with the addition of speech recognition and text to speech capabilities, models could conceivably communicate with the live center controllers. They can also be used as pilots - eliminating the need for pseudo-pilots to take part in evaluations, as is typically done. All of this means far less manpower is necessary in enacting a HITL of this type.

B. *Benefits of the CITL*

Cognitive models, embedded within a simulation environment, allow for macro level system results that are based on, among other things, micro level cognitive data of a type typically only gained in human-in-the-loop evaluations. This then allows us to evaluate the impact of a new technology on something much closer to a system-wide scale. Perhaps just as importantly, these types of evaluations can happen early in the design process (even before working prototypes of the concept to be tested are developed). Because the models identify potential problems with a design, this also means that fairly early in concept development it would be possible to either make changes for a more effective result or determine that the concept is not one that is worth pursuing. The result is fewer design iterations and the ability to test with live controllers and pilots nearer to a concept's final form. The CITL is, of course, not without drawbacks. Model Human Processor (MHP) styled cognitive models assume expert, error free behavior and will often represent the visual system in a very rudimentary way. They are also very rigid ways of representing a decision making process that may actually be quite flexible.

IV. EDGAR

In the fall of 2006 MITRE began developing its own en route controller model, Edgar, with the ultimate goal of creating an agent that could a) replicate the most common en route controller behaviors and b) be embedded as an autonomous agent within our real time, en route simulation environment. To that end, we began to look for a cognitive architecture - in this case a software package - that would meet our requirements which included:

- Relatively easy to learn and use
- Provisions to link the architecture to an outside simulation environment
- Providing a basic set of cognitive resources including a visual system, working memory, long term memory, and procedural task execution
- Ability to multitask
- Temporal and mental workload estimates

A. *Fidelity*

Most of these requirements are answers to the implicit question of fidelity. That is, for our purposes, at what level of detail is it necessary to replicate human behavior? Is it required that the model specify down to the level of individual neurons, or is it sufficient to work at the unit-task level? Consider the image shown in Figure 2. The deer depicted in the image is clearly being reflected in the stream. Even without the context of the rest of the image, the reflection, drawn with much less detail, is still clearly recognizable. If the goal is only to communicate the image of a deer, either rendering would do. And, if the goal is to do so efficiently, the reflection is actually preferable. This is a reasonable, if imprecise, analogy for the level of detail at which we found it necessary to model en route controller behavior. Our goal is not to determine how individual neurons perform, but rather to quantify how a controller performs. So, if the emergent behavior of the controller can be adequately represented with less detail and effort than is, for example, required for neurological models, it is preferable in an applied setting to use a lower fidelity architecture which will presumably be easier to learn, build, modify, and implement.

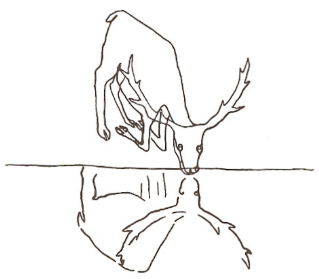


Figure 2. Fidelity [5]

In comparing the fidelity required to the fidelity provided by a large variety of cognitive architectures we found a subset of architectures that addressed our needs including CORE [22], APEX [11], ACT-R [2], and GLEAN [15]. Ultimately, we chose GLEAN. Among the major determinants were the language GLEAN is built in (it needed to be amendable to our lab infrastructure), the ease of modeling, and the fact that GLEAN is based on MHP-style cognitive modeling theory with which we were already experienced.

B. *GLEAN Cognitive Architecture*

GLEAN formalizes a cognitive modeling technique known as GOMS (Goals Operators Methods & Selection Rules), a derivative of the Model Human Processor. "Paper and pencil" versions of GOMS have been used very successfully at MITRE for a wide variety of projects including analyses of tower, en route, and TRACON controllers, analysis of pilots using advanced delegated separation tools, and analyses of Traffic Managers using Next Generation Air Transportation System (NextGen) style automation [8, 9]. GLEAN provided an avenue to continue this same type of analysis, but on a much larger scale. Kieras [15] describes GOMSL/GLEAN in the following way:

"GOMSL runs in a simple cognitive architecture, implemented in the GLEAN simulation environment, that represents the assumed mechanisms and capabilities of the human information processing system. The cognitive architecture consists of a set of human-like processors and output capabilities, each constrained according to the limitations of the human cognitive system. "

According to Kieras, processors and output devices, include:

"Auditory processor accepts either speech or sound inputs and makes them available to the cognitive processor, where they are tested or waited for by auditory operators. The visual processor accepts visual events that take the form of an object appearing with specified properties, or a change in a visual property of an existing visual object. Visual input is either searched or waited for with visual operators. Vocal output is produced when a vocal operator is executed, specifying an utterance to be produced, which is sent to the device or another simulated human auditory processor. Manual output takes many different forms depending on the type of movement specified by a manual motor operator. The architecture contains certain memory structures. Each perceptual processor has its own working memory. The cognitive processor's working memory has two partitions: an object store and a tag store. The tag store represents the conventional human working memory, and contains tags (labels) each with an associated value. "

It is also worth noting that GLEAN is multi-threaded, which allows for simulated human multitasking, increasing the accuracy of the task time prediction.

C. *Model Building*

A representative controller model executes a set of controller tasks. The tasks included in that set determine, in part, how thoroughly the whole of controller behavior is modeled. The tasks selected for Edgar currently include: accepting handoffs, offering handoffs, maintaining spacing on a flow, identifying and resolving pairwise conflicts, and implementing Miles-in-Trail (MIT) restrictions.

Each of these tasks was formalized using a form of Cognitive Task Analysis (CTA) known as Hierarchical Task Analysis (HTA). HTA both defines the knowledge required for completing a task and presents those tasks in the order in which they must be executed [16]. As an example, the HTA for the offer handoff task (also known as initiate handoff or exit) is shown in Figure 3. As shown in the figure, the controller begins by monitoring the Display System Replacement (DSR) for aircraft in position to be handed off. The task proceeds through the controller completing the necessary data entry and verbalizing the handoff to the pilot. It includes activities that controllers often complete to facilitate remembering like adjusting the length of the leader line as a reminder of which aircraft have checked in on the frequency. HTA, informed by literature review and controller interviews, was conducted for each of the previously described tasks.

Once the HTA has been completed for a given task, it can be translated into GLEAN syntax (GOMSL). This process involves converting each cell of the above HTA into a goal or step in a method. The GOMSL syntax for the Initiate Handoff HTA is shown in Figure 4. Notice that some cells of the HTA

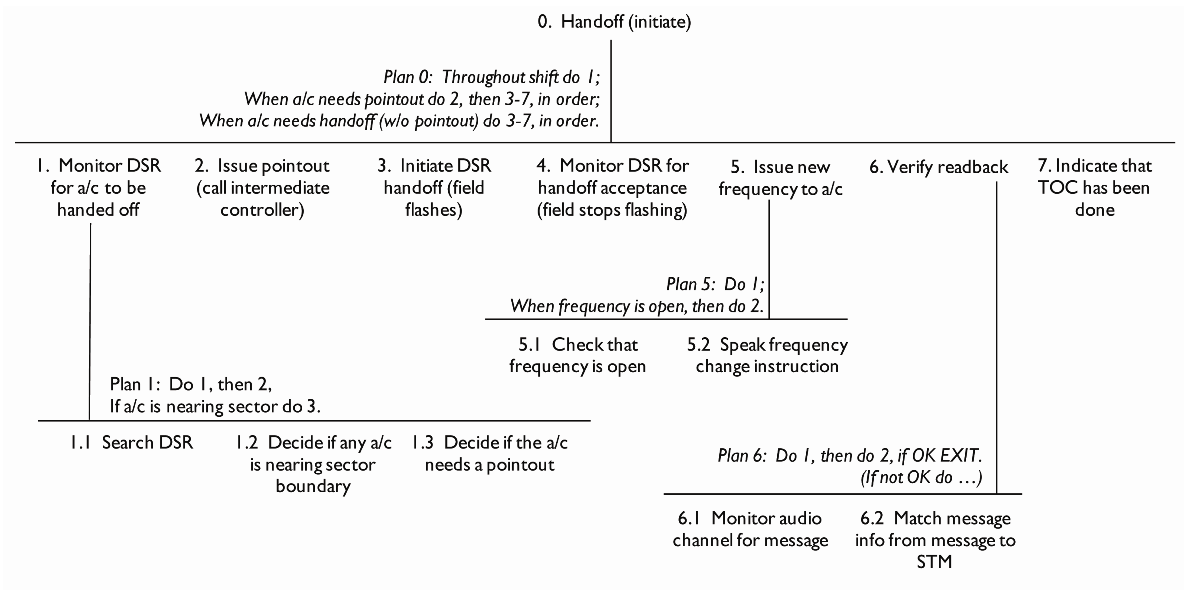

0. Handoff (initiate)
Plan 0: Throughout shift do 1; When a/c needs pointout do 2, then 3-7, in order; When a/c needs handoff (w/o pointout) do 3-7, in order.
1. Monitor DSR for a/c to be handed off
2. Issue pointout (call intermediate controller)
3. Initiate DSR handoff (field flashes)
4. Monitor DSR for handoff acceptance (field stops flashing)
5. Issue new frequency to a/c
6. Verify readback
7. Indicate that TOC has been done
Plan 5: Do 1; When frequency is open, then do 2.
5.1 Check that frequency is open
5.2 Speak frequency change instruction
Plan 1: Do 1, then 2, If a/c is nearing sector do 3.
1.1 Search DSR
1.2 Decide if any a/c is nearing sector boundary
1.3 Decide if the a/c needs a pointout
Plan 6: Do 1, then do 2, if OK EXIT. (If not OK do …)
6.1 Monitor audio channel for message
6.2 Match message info from message to STM

Figure 3. HTA for Handoff Task

```
Method_for_goal: Accept_Handoff Loop
        Step. Delete <current_ac>; Delete <next_ac>.
        Step Begin. Look_for_object_whose Handoff_Status is "Eligible_Incoming" and_store_under <current_ac>.
        Step log.  Log <current_ac>.
        Step. Decide: If <current_ac> is Absent Then Goto Fine.
        Step. Accomplish_goal: Get Accept_Handoff_Data.
        Step. Accomplish_goal: Accept_Handoff Data_Entry.
        Step. Accomplish_goal: Accept_Handoff Data_Block_Cleanup.
        Step. Accomplish_goal: Say Hello.
        Step. Delete <current_cid>; Delete <current_acid>; Delete <carrier>.
        Step. Delete <flight_id>; Delete <ho_status>; Delete <next_ac>.
        Step. Goto Begin.
        Step Fine. Return_with_goal_accomplished.

Method_for_goal: Get Accept_Handoff_Data
        Step 1. Store CID of <current_ac> under <current_cid>; Log <current_cid>.
        Step 2. Store ACID of <current_ac> under <current_acid>; Log <current_acid>.
        Step 3. Store Airline of <current_ac> under <carrier>; Log <carrier>.
        Step 4. Store Flight_ID of <current_ac> under <flight_id>; Log <flight_id>.
        Step 5. Store Handoff_Status of <current_ac> under <ho_status>; Log <ho_status>.
        Step 6. Return_with_goal_accomplished.

Method_for_goal: Accept_Handoff Data_Entry
        Step 1. Type_in <current_acid>.
        Step 2. Verify Correct.
        Step 3. Keystroke Enter.
        Step 4. Return_with_goal_accomplished.

Method_for_goal: Accept_Handoff Data_Block_Cleanup
        Step Begin. Type_in "6/2 ".
        Step. Type_in <current_acid>.
        Step. Verify Correct.
        Step. Keystroke Enter.
        Step. Log Leader_Length of <current_acid>; Log Leader_Direction of <current_acid>.
        Step. Return_with_goal_accomplished.

Method_for_goal: Say Hello
        Step 1. Store Airline of <current_ac> under <carrier>; Store Flight_ID of <current_ac> under <flight_id>.
        Step 2. Recall_LTM_item_whose Abbrev is <carrier> and_store_under <airline_name>.
        Step 3. Keystroke Push_to_Talk.
        Step 4. Speak Call_Sign of <airline_name>.
        Step 5. Speak <flight_id>.
        Step 6. Speak "Jacksonville Center, good day".
        Step 7. Keystroke Release_Push_to_Talk.
        Step 8. Return_with_goal_accomplished.
```

Figure 4. GOMSL Syntax for Accept Handoff Task

are steps of a larger goal (e.g., monitoring the DSR) while others are the basis of the goal itself (e.g., Accept_Handoff Data_Entry). It is sometimes the case that the HTA will not provide all details necessary to complete the GLEAN model. For example, the HTA calls out the "Initiate DSR Handoff" task but does not specify the keystrokes necessary execute that task. In the model, those steps must be enumerated, and this will often require further consultation with experts. The complete method for initiate handoff data entry is shown under "Initiate Handoff Data Entry."

D. *Model Strategy for Conflict Resolution*

Two of the more interesting tasks to model were spacing on a flow (maintaining the prescribed horizontal separation minima for aircraft following the same path) and pairwise conflict resolution (maintaining the prescribed horizontal and vertical separation minima for aircraft on different paths). Because these tasks are two of the more difficult controller tasks and the resulting task models provided insight into controller cognition, they merit further discussion. There were two important principals, both culled from the literature and applied to the modeling effort, for replicating the process by which controller's resolve these flow and pairwise conflicts: Long Term Working Memory (LTWM) and satificing using hierarchical decision making.

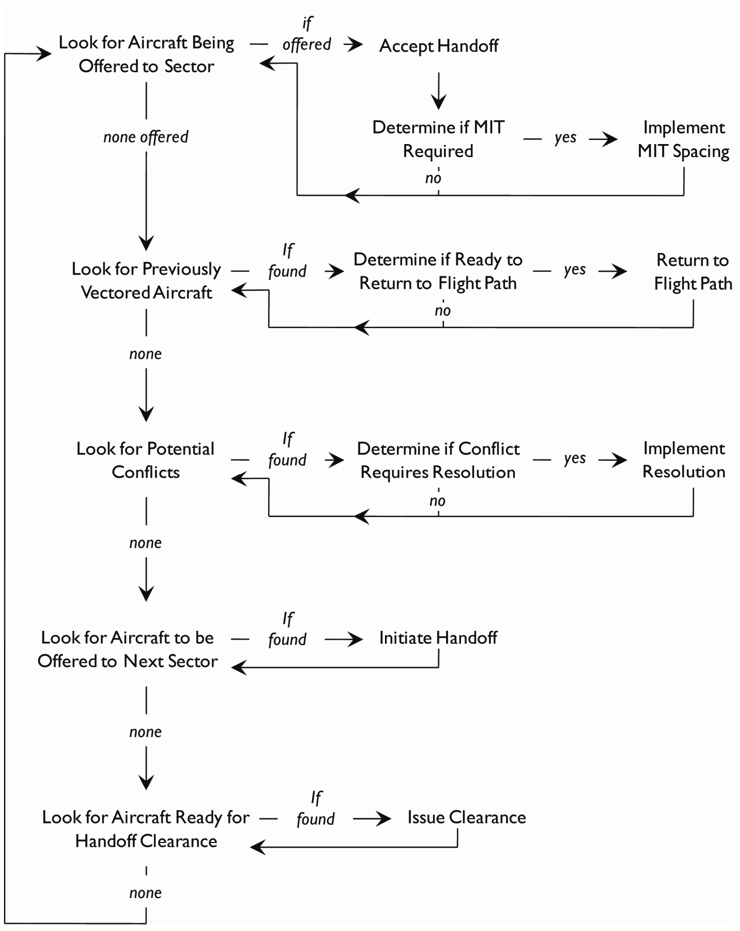

Look for Aircraft Being Offered to Sector
if offered
Accept Handoff
Determine if MIT Required
yes
Implement MIT Spacing
no
none offered
Look for Previously Vectored Aircraft
If found
Determine if Ready to Return to Flight Path
yes
Return to Flight Path
no
none
Look for Potential Conflicts
If found
Determine if Conflict Requires Resolution
yes
Implement Resolution
no
none
Look for Aircraft to be Offered to Next Sector
If found
Initiate Handoff
none
Look for Aircraft Ready for Handoff Clearance
If found
Issue Clearance
none

Figure 5. Edgar's Control Loop

Controllers have the ability to store an exceptional amount of information in short term, or working, memory. While the average person can store about seven plus or minus two chunks of information in working memory [17], the expert controller has been found to store up to thirty pieces of information about a set of aircraft [4]. This capacity, which allows the controller to process large amounts of information, is extraordinarily useful as, beyond processing large amounts of "on-screen" information, the controller must analyze the traffic display according to knowledge stored in memory. Learned knowledge, or knowledge stored in Long Term Memory, may include information about aircraft, effects of altitude of performance, Standard Operating Procedures and Letters of Agreement for the sector, a large number of heuristics for determining separation, and a repository of knowledge based on prior experience with any number of different conflict geometries.

The controller's ability to retain and quickly recall this information from memory may be accounted for by the theory of Long Term Working Memory [7]. The theory of Long Term Working Memory posits that experts in a given domain are able to enhance working memory through the use of enhanced retrieval structures. In short, the controller's knowledge of the domain is stored in such a way that it is easily retrievable when tied to cues in working memory. In practice, this effect has been seen in chess, medical diagnosis, short order cooking, and flight planning. Under this theory, controllers become highly efficient processors of information.

There is also a corollary capability that a controller must develop: a controller must also be an expert "forgetter". Even with the expanded capacities that Long Term Working Memory provides, there are still capacity limits, and new information will eventually overwrite or interfere with old information. Further, because air traffic control is a highly dynamic domain, there are distinct disadvantages to holding information in memory for long periods of time. For example, the current speed of the aircraft may be no indication of the future speed of the aircraft. The DSR also obviates the need to store information in memory for very long. It is clearly advantageous to the controller to rely on perfect knowledge in the world (the DSR) rather than imperfect knowledge in the head [19]. Other researchers have speculated that this type of forgetting exits in air traffic control [13] and theories like functional decay [1] have given algorithmic descriptions of accelerated forgetting among experts.

Functional Decay and Long Term Working memory have had important impacts on our controller agent. First, Edgar "forgets" as soon as the information is no longer needed. This typically means information will be held in working memory for, at most, 10 seconds. Second, rather than approaching conflict problems mathematically, our model relies on many large sets of decision rules, stored in Long Term Memory and accessed via cues in working memory, just as a human controller would. Further, Edgar employs these decision rules under a strategy of satisficing (as does the controller), which entails finding a suitable solution to a problem that be sub-optimal. This strategy allows a controller to quickly determine whether a potential conflict will in fact result in a loss of separation. This, it would seem, is a reasonable tradeoff as a) the goal is to find a solution that will maintain separation and b) trading speed for a less than optimal solution accomplishes that goal safely.

E. Overview of Model Execution

Edgar executes all the modeled tasks according to a control loop, much as a human controller would. In the control loop, Edgar begins by looking for aircraft in position to be accepted into the sector. Edgar then looks for aircraft for which a previously issued vector or altitude needs to be resolved. Edgar subsequently looks for and addresses any new conflicts (including pairwise and spacing on a flow). Edgar ends the loop by looking for aircraft in position to be handed off to the next sector after which the loop starts again. A flow chart representing both the control loop and some aspects of Edgar's decision making process can be found in Figure 5.

F. Model Outputs

When running a CITL evaluation, Edgar is instantiated in the sector or sectors of interest in our en route simulation environment. As Edgar runs, it generates a log file or trace of the controller model's activities. The model trace contains information about statement execution time, visual system use, and storage to and retrievals from working memory among other times. An annotated example of one line of output from a model is shown in Figure 6.

When a trace is generated, it is fed through an analysis tool called CADRE (Cognitive Agent Data Reduction Effort). CADRE identifies all the methods executed, the execution time for each method, and the working memory load for each method.

Based on the model output, statistics are generated to measure the controller's workload. Workload is characterized by the mental difficulty and time available to do the work. Time available, or temporal workload, is a relatively straightforward measure. Using the statement time we can observe how long it took Edgar to accomplish a set of tasks. Where appropriate, that time can be compared to the amount of time available to accomplish those tasks. However, as CADRE is most often used to compare efficiency of the controller with and without some new tool (for example, the amount of time to complete a handoff with data link versus over the voice channel), it is typically more interesting to compare total task time across tasks than to the total time available for completion of tasks.

Mental workload is measured in a different manner. CADRE assesses the mental difficulty of the work based on what may loosely be described as the amount of information the controller is dealing with during any given 50 millisecond cycle of the model's execution. More formally, it is the total number of chunks stored in the model's working memory resource. Each time a piece of information is stored or deleted from the working memory resource, CADRE notes this change and updates the count of the total number of items stored in working memory.

This count can then be compared against known working memory thresholds (as previously discussed, the average person can hold about seven chunks of information in working memory). This limited capacity means that working memory is often a bottleneck in problem solving. It is also a good indicator of the cognitive difficulty of the work. Interestingly, such quantitative and objective measures of working memory load and can only be measured with a model (a human cannot tell you how much information is stored in working memory). As such, it illustrates another way in which models are powerful - they open (slightly) the black-box process of cognition so that it may be observed. So, not only do we know the magnitude of the mental workload, but, in cases where the work is found to be particularly difficult, the exact cause of that difficulty.

V. Validation Exercise

When cognitive models are validated, it is almost always done by having a model and human complete the same set of tasks and subsequently examining how well the model performance conforms to human performance. A similar method is employed here, however, rather than running human participants for the express purpose of testing the model, we gathered data from the literature.

Human performance is a broad term. In the course of evaluating a model, any number of aspects of performance may be examined. Among those are: visual scan, decision accuracy, learning rates, and perceptual motor skill. In the coming years we will likely examine Edgar's ability to replicate human performance in most of these areas, but currently we are focused on temporal workload and task times.

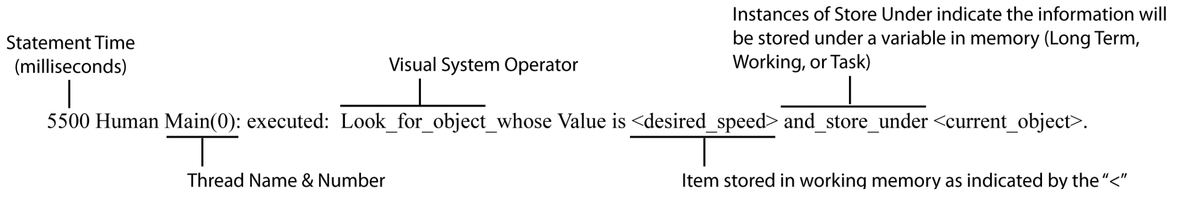
One of key constraints on increasing NAS capacity is the controller. The controller simply has a finite amount of time to complete tasks. Increases in demand will ultimately lead to a situation in which the amount of time available is exceeded by the total time necessary to complete the tasks at hand. As the time available to complete those tasks cannot be changed, an important component of increasing NAS capacity will be decreasing task time.

As part of NextGen, automation will be provided to the controller that enables task time reduction. In evaluating and designing NextGen automation tools, one of the more important goals is quantifying exactly by how much task times are reduced and, as a result, how much capacity is gained in the NAS.

So, in as much as Edgar can provide immediate benefit in evaluating questions of controller efficiency and productivity, verifying that Edgar is able to replicate controller task times is a high priority for this research. To do so, CADRE was used to analyze task times generated by Edgar for a set of tasks for which we also had human data.

In some cases, the way in which Edgar completed a task differed from the way the task was executed in the literature. Where those differences existed, Edgar was, for the purposes of the validation exercise, modified to execute the task as defined in the reference material. Although Edgar's task time may vary slightly run-to-run depending on things like the length of the clearances being issued, for this analysis we ran Edgar only once for each task. As such, Edgar produces a single value for comparison to the observed data culled from the literature.

Figure 6. Annotated GLEAN Output


Statement Time (milliseconds)
Visual System Operator
Instances of Store Under indicate the information will be stored under a variable in memory (Long Term, Working, or Task)
Thread Name & Number
Item stored in working memory as indicated by the "<"

```
5500 Human Main(0): executed: Look_for_object_whose Value is <desired_speed> and_store_under <current_object>.
```

A. Validation Results

In the first test, we compare the model and human time required to complete the transfer of communication and initial contact tasks using data from a data communications study conducted by the Federal Aviation Administration (FAA) Free Flight office [3]. It should be noted that because the time to complete the flight plan review component of the initial contact task was not included in the human times for initial contact, we removed it from the model time. Three comparisons were made. Two of these tasks required the controller to use the voice channel to issue the relevant clearance and in the third the controller used data communications rather than the voice channel to issue the relevant clearance. The results are shown in Figure 7.

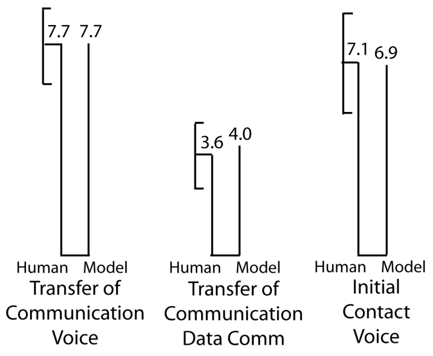


Figure 7. Time in Seconds to Complete Specified Task

The greatest discrepancy between the model prediction and the observed average was found for the Transfer of Communication Data Comm task at 400 ms. This result is both within the standard deviation (depicted by the error bars) and, from a standpoint of practical application, well within the range of acceptable results. The model prediction for Initial Contact Voice performed slightly better, within 200 ms of the observed data and, again, well within the standard deviation. The average task time for Transfer of Communication Voice was identical to the model prediction.

It is very important that Edgar provide accurate temporal predictions for the Transfer of Communication and Initial Contact tasks because they both occur frequently (once for each aircraft in the sector). These task times are driven by data entry and vocal utterances, both of which are known to be much easier to provide accurate temporal predictions for than are other aspects of human performance. Providing accurate task times for more nuanced decision making tasks and decision making tasks which demand considerably more of the cognitive system is a more difficult challenge. As a result, we chose one component of a task which fits this description for another validation exercise.

The conflict resolution task includes, at a high level, conflict identification, generation of conflict resolution, and issuance of that resolution. While issuance of the resolution falls into the bin of tasks we would expect to provide quite accurate task times for, both the conflict identification and resolution are expected to be more difficult to accurately predict temporally. Further, conflict identification is implemented in Edgar in such a way that task time varies based on the conflict geometry.

In a paper on the conflict identification process, Rantanen and Nunes [20] provide a series of conflict identification times which, as with Edgar, vary based on the geometry of the conflict. According to Rantanen and Nunes, the length of time required to identify the conflict will depend on the differential speed, heading, and altitude of the aircraft involved in the conflict. The shortest conflict identification times should be for aircraft that have altitudes that will exceed separation minima at the conflict intersection, as altitude will be evaluated first by the controller. If the delta in altitude will be within the separation minima, the controller must further evaluate the conflict according to heading. If the aircraft paths are deemed to intersect, the controller must finally evaluate the conflict according to speed. Clearly, the earlier this decision hierarchy is exited, the quicker the conflict is identified.

Edgar, informed by our task analysis and controller interviews, operates in much the same way. However, because heading is identified as an initial pattern recognition in the conflict identification process, our hierarchy begins with heading and is subsequently followed by altitude and speed rather than beginning with altitude. Likewise, although the magnitude of the conflict angle would had an impact on the time to identify the conflict in Rantanen and Nunes, it has no effect in Edgar. As such, the results shown in Figure 8 compare only the results for two aircraft that have a heading difference of zero degrees (that is, two aircraft on the same flight path).

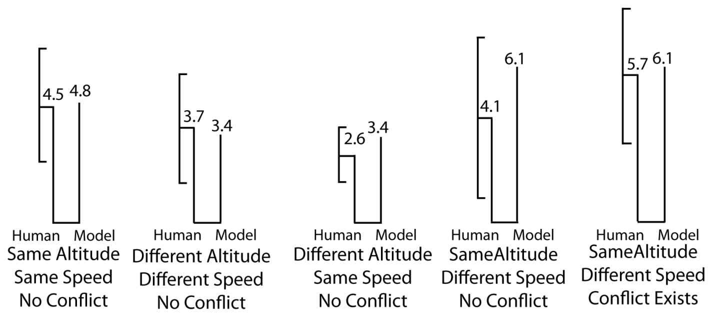


Figure 8. Time in Seconds to Identify Conflict

When the model performance is compared to that of the performance of participants in Rantanen and Nunes, we again find all results within the standard deviation for human data. However, it is worth noting that where subjects in the Rantanen and Nunes experiment showed higher identification times when a conflict actually existed, Edgar does not replicate the result. This is because the point at which Edgar exits the conflict identification loop determines whether or not the conflict exists, whereas a human may further perform a mental validation of the result.

Thus far for the tasks analyzed, Edgar has performed quite well. Further research needs to be done on task time for the conflict resolution task as well as for additional tasks to be added to Edgar's repertoire in the future like the ability to handle point-outs and execute merging. It will also be necessary to evaluate the impact of multitasking, which Edgar can do, on total performance time.

Most evaluations of controller task time consider the task in isolation. However, controllers often accomplish several tasks at once. For example, while issuing a voice clearance to an aircraft, the controller can scan for conflicts or aircraft in position for handoff. So if an observer were to add up the task times for all of the tasks accomplished and consider that total execution time, it would often exceed the actual total execution time of the controller, who is able to accomplish some tasks (or portions of some tasks) simultaneously.

VI. SUMMARY

As Edgar becomes more capable, we will continue to test its predictive power. This will include not only its ability to estimate task times, but also mental workload and emergent behavior. In the mean-time, we have a good initial en route controller model for use in CITLs which gives us the capability to evaluate NextGen concepts on a much larger scale then would be possible with live controllers and pilots.

Cognitive Models are a long way from being able to replicate human behavior so accurately that real controllers and pilots are no longer needed in human-in-the-loop evaluations. And there is no intention of using Edgar to stop the steady stream of pilot and controllers participating in evaluations at MITRE. As long as there are controllers and pilots in the system, we will always consider their input on the impacts of new procedures and technologies above that of a cognitive model. After all, there really is no replacement for the real thing.

But, if we can use models early in the design process and in large, CITL evaluations, they will be able to supplement controller feedback with insights into how a new technology will affect not just a sector, but the entire NAS. In the long term, our hope is that Edgar, a rough carbon copy of human behavior, can help to get tools into the hands of pilots and controllers that will help them improve safety and efficiency in the NAS.

ACKNOWLEDGMENTS

The authors would like to thank the various managers, computer scientists, and air traffic controllers who aided in the development of Edgar including: Charlie Bailey, Bill Bateman, Bob Bearer, Ed Brestle, Gary Bogle, Chris DeSenti, Alfreda Gipson, Urmila Hiremath, Dr. David Kieras, Keith Knudsen, Chris Magrin, Scott Mayer, Dien Nguyen, Elliott Simons, Mike Tran, Zach Eyler-Walker, Scott Wood, and Soar Tech.

REFERENCES

[1] Altmann, E. M., & Gray, W. D. (2002). Forgetting to remember: The functional relationship of decay and interference. Psychological Science, 13(1), 27-33.

[2] Anderson, J. R., & Lebiere, C. (1998). The Atomic Components of Thought. Mahwah, NJ: Erlbaum.

[3] Bennett, Michael (2003). CPDLC Benefits Story. FAA Free Flight Office, Unpublished.

[4] Bisseret, A Mémoire opérationelle et structure du travail Bulletin de Psychologie 1970 XXIV 280-294 English summary in Ergonomics, 1971, 14, 565-570.

[5] Calder, Alexander (1967), Fables of Aesop, According to Sir Roger L'Estrange New York: Dover Publications.

[6] Card, Stuart; Thomas P. Moran and Allen Newell (1983). The Psychology of Human Computer Interaction, Lawrence Erlbaum Associates. ISBN 0-89859-859-1.

[7] Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. Psychological Review, 102(2), 211-245.

[8] Estes, S.L. (2005). Arriving from Delphi at O'Hare: Predictive Cognitive Engineering in the O'Hare Modernization Project and Beyond. In Proceedings of the 49th Annual Meeting of Human Factors and Ergonomics Society, Orlando, FL

[9] Estes, S.L., Masalonis, A.J. (2003). I See What Your Thinking: Using Cognitive Models to Refine Traffic Flow Management Decision Support Prototypes. In Proceedings of the 47th Annual Meeting of Human Factors and Ergonomics Society, Denver, CO.

[10] Fitts, Paul (1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology, volume 47, number 6, June 1954, pp. 381-391.

[11] Freed, M., Matessa, M., Remington, R., and Vera, A. (2003). How Apex Automates CPM-GOMS Proceedings of the 5th International Conference on Cognitive Modeling. Bamberg, Germany.

[12] Gronlund, S. D., Dougherty, M. R. P., Ohrt, D. D., Thompson, G. L. & Bleckley, K. M. (1997). The Role of Memory in Air Traffic Control. (Report No. DOT/FAA/AM-97/22) Washington, DC: Federal Aviation Administration.

[13] Kieras, David (1988). "Towards a practical GOMS model methodology for user interface design". in Martin Helander. Handbook of Human-Computer Interaction. Amsterdam, The Netherlands: Elsevier Science Publishers. pp. p135-157. ISBN 0-444-88673-7.

[14] Kieras, D., (1996) A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN4, University of Michigan.

[15] Kirwan, B. & Ainsworth, L. K. (Eds.) (1992). A guide to task analysis. London, UK: Taylor & Francis.

[16] Miller, G. A. (1956). The Magical Number Seven Plus or Minus Two: Some limits on our capacity for processing information. Psychological Re view, 63, 81-97

[17] Newell, A. (1973). "You can't play 20 questions with nature and win: Projective comments on the papers of this symposium". In W. G. Chase (ed.), Visual Information Processing. New York: Academic Press.

[18] Norman, Don (2002), The Design of Everyday Things. New York: Basic Books.

[19] Rantanen, E. M., & Nunes, A. (2005). Hierarchical conflict detection in air traffic control. International Journal of Aviation Psychology, 15(4), 339-362.

[20] Rosenbloom, P. S., Laird, J. E., & Newell, A. (1993) The Soar papers: Research on Integrated Intelligence. MIT Press, Cambridge, MA.

[21] Tollinger, Irene, Richard L. Lewis , Michael McCurdy , Preston Tollinger, Alonso Vera, Andrew Howes, Laura Pelton, Supporting efficient development of cognitive models at multiple skill levels: exploring recent advances in constraint-based modeling, Proceedings of the SIGCHI conference on Human factors in computing systems, April 02-07, 2005, Portland, Oregon, USA

AUTHOR BIOGRAPHY

Steven L Estes lives in Savannah, Georgia. He holds a BA in History (1996) from the University of Georgia in Athens, Georgia and an MA in Human Factors and Applied Cognition (2002) from George Mason University in Fairfax, Virginia.

He is currently a Lead Human Factors Engineer at the MITRE Corporation's Center for Advanced Aviation System Design in McLean, Virginia. Prior to working for MITRE, he was employed as a human factors engineer at Gulfstream Aerospace. Publications include the book chapter the book chapter "Macrocognition in systems engineering: supporting changes in the air traffic control tower", published in the book Naturalistic Decision Making and Macrocognition (Burlington, VT: Ashgate Publishing Company, 2008). Research interests include: cognitive engineering, human computer interface design, human decision making, and human factors in the aviation domain.

Mr. Estes is a member of the Cognitive Science Society and the Human Factors and Ergonomics Society.

Craig A. Bonaceto lives in Chelmsford, Massachusetts. He holds a dual BS degree in computer science and psychology (2001) from Rensselaer Polytechnic Institute in Troy, New York.

He is currently a Senior Information System Engineer at the MITRE Corporation's Command and Control Center in Bedford, Massachusetts. Prior publications include the book chapter "A survey of the methods and uses of cognitive engineering", published in Expertise Out of Context (New York, New York: Lawrence Erlbaum Associates, 2007), and the book chapter "Macrocognition in systems engineering: supporting changes in the air traffic control tower", published in the book Naturalistic Decision Making and Macrocognition (Burlington, VT: Ashgate Publishing Company, 2008). He applies Cognitive Engineering methods to improve the design of Enterprise Systems in air traffic control and military command and control. He has also performed research on human "mental models" to improve Human-System Integration.

Mr. Bonaceto is a member of the Cognitive Science Society and the International Council on Systems Engineering (INCOSE).

Kevin Long currently lives in Great Falls, Virginia. He holds a BA in Sociology (2006) from The Catholic University of America in Washington D.C. and is currently pursuing a MA in Human Factors Applied Cognition from George Mason University in Fairfax, Virginia.

He is currently employed as a Human Factors Engineer at the MITRE Corporation's Center for Advanced Aviation Systems Development in McLean, Virginia. Research interests include human factors in the aviation domain, specifically situation awareness and workload, and human computer interface design.

Mr. Long is a member of the Human Factors and Ergonomics Society (HFES).

Scott H. Mills lives in Arlington, Virginia. He holds a BA degree in psychology (1987), an MS in experimental psychology (1991), and a PhD in experimental psychology (1995) from the University of Oklahoma in Norman, Oklahoma.

He is currently a Lead Multi-Discipline Systems Engineer at the MITRE Corporation's Center for Advanced Aviation System Design in McLean, Virginia. Prior to working for MITRE, he was a Senior Human Factors Engineer at SBC Technology Resources in Austin, Texas and an Engineering Psychologist at the FAA Civil Aerospace Medical Institute in Oklahoma City, Oklahoma. Research interests include human factors in aviation and human computer interface design.

Dr. Mills is a member of the Human Factors and Ergonomics Society.

Frank M. Sogandares lives in the Northern Virginia suburbs of Washington, DC. He holds a BA in physics (1983) from Hendrix College in Conway, Arkansas and a PhD in physics (1991) from Texas A&M University in College Station, Texas.

He is currently a Principal Engineer with the MITRE Corporation's Center for Advanced Aviation System Design in McLean, Virginia. His interests are distributed computation, languages, and the application of these to real-world problems.