Better Model-Based Analysis of Human Factors for Safe Aircraft Approach

Joseph Krall, Tim Menzies *Member, IEEE*, Misty Davies *Member, IEEE*

***Abstract*—How can analysts better reason about the complex interactions between humans and machines in safety critical systems? For example, consider the complexities arising when a large commercial aircraft makes a runway approach. At that time, pairs of pilots interact with both each other and a large number of intricate on-board automated systems. Any model of that kind of pilot interaction is inherently complex and, therefore, inherently hard to commission, debug, and study.**

We advocate exploring complex models by combining data miners (to find a small set of most critical examples) and of multi-objective optimizers (that focus on those critical examples). An example of such a combination is the GALE optimizer that intelligently explores thousands of scenarios by examining just a few dozen of the most informative examples. GALE-style reasoning enables a very fast, very wide-ranging exploration of behaviors, as well as the effects of those behaviors' limitations.

***Index Terms*—Human Factors, Cognitive Modeling, Multi-objective Optimization, Active Learning**

I. Introduction

There are many advantages of a model-based approach to human factors. Traditional human-in-the-loop experimental case-studies are expensive, time consuming, and difficult to reproduce. Model-based conclusions, on the other hand, are reproducible and verifiable (just run the model again). Another advantage of the model-based approach is that models can simulate real world behavior much faster than real-time; thereby enabling an extensive evaluation of more options than slow-time real-world case studies.

In theory, complex models can be analyzed via multi-objective optimizers by running them across many CPUs. In practice, that CPU may not be available. For example one of us (Davies) regularly analyzes a model that needs 30 weeks of CPU time. For high priority issues in need of urgent resolution, then this 30 weeks of computer time can be achieved in five days of parallel execution on NASA's supercomputers. However, researchers can usually access a small fraction of that CPU. For example, if there has been some incident on a manned space mission, NASA enlists all available CPU time for "damage modeling" (which is a large series of "what-if" queries to assess potential impacts). At those times, researchers can access zero CPU for any other purpose.

To address this problem in other domains, we have proposed a new optimization method, called GALE [1]–[3], that focuses on a small number of most informative examples. Hence, GALE explores only a few dozen examples rather than the thousands (or more) used by traditional methods [1], [3]. This simplifies and improves our ability to reason about complex cognitive models.

In practice, GALE runs much faster than traditional optimizers. Standard optimization algorithms such as NSGA-II [4] require 3000 to 5000 evaluations to explore the pilot simulator that is the large study of this paper. GALE, on the other hand, performs the same task using 25 to 50 evaluations [2]. In practice, this has tremendous practical implications. When generating conclusions from a randomizing optimizer such as GALE or NSGA-II, it is important to check that the conclusions hold in multiple repeats (say, 20 repeats). This number of evaluations is a critical indicator of runtime in optimizers when the model is complex. For example, a study of the pilot simulator (replicated 20 times) takes 1.5 and 100 hours for GALE and NSGA-II, respectively [2].

The algorithms behind GALE have been presented previously [1]–[3]. Those prior reports focused on runtimes and did not explore the analysis implications of GALE. The core contribution of this paper is a case study on how GALE-style reasoning assists in the analysis of cognitive modeling. This paper takes a large model of pilot cognition (CDA [8]–[12]) and explores in detail how GALE's conclusions relate to pilot cognitive workloads and safe aircraft operation within the context of that model. As shown in §III, (1) GALE offers many novel insights into complex cognitive models; (2) other methods would be so slow to run that it might be impractical to find those insights without GALE.

The rest of this paper is structured as follows. Figure 2 lists the frequently used acronyms in this paper. §II motivates this paper with a review on how cockpits are becoming increasingly more complex. An effective search of the models requires the utilization of automated tools such as the multi-objective objective evolutionary algorithms (MOEAs) discussed in §III. These models are intricate and take time to run. One such MOEA is our GALE tool discussed in §III-B which, in §IV is applied to a large simulation of pilots flying a plane. We show that GALE can answer cognitive research questions like:

- RQ1: Given a limited maximum human task load capacity, what are the effects to safety assurance when a pilot's workload exceeds his or her capacity?
- RQ2: What are the effects of changing how much pilots rely on automation?
- RQ3: What are the effects of changing pilot policies for monitoring and overlooking flight procedures?

Joseph Krall is a postdoctoral research fellow at LoadIQ, Reno, Nevada; e-mail: kralljoe@gmail.com

Tim Menzies is with the Computer Science department, NcState University; email: tim.menzies@gmail.com.

Misty Davies is with the Intelligent Systems Division, NASA Ames Research Center, CA, USA; e-mail: misty.d.davies@nasa.gov.

Fig. 1a: Haviland Moth,1936 [5].

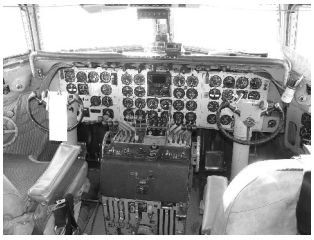

Fig. 1b: DC7, 1953 [6].

Fig. 1c: Airbus 380, 2005 [7].

Fig. 1: The evolution of airplane cockpit complexity from 1936 to 2005.

II. Motivation

This work is motivated by the growing complexity of models. For example, consider a model of pilot behavior. As shown in Figure 1a, a cockpit in 1936 was little more than a starter, a yoke, a throttle, pedals, and about a dozen gauges that the pilot could use to monitor airspeed, pressures, engine speed and altitude. A pilot model for that kind of plane would be relatively simple.

By 1953, pilots had access to much more information. Civil transport aircraft now included redundant systems, including a copilot, and the other machinery as shown in Figure 1b. Concepts of operation had to clearly define the roles and the responsibilities of the pilot and copilot. For these kinds of planes, pilot simulation models would be more complex.

By 2010, aircraft had become computers with wings; e.g. the Airbus 380 cockpit of shown in Figure 1c. Autopilots allow the pilot to turn over much of the route planning and control to automation on the flight deck, and instead pilots can now take a more passive monitoring role. While automation has certainly helped to reduce the number of aviation accidents [13], the kinds of errors that lead to accidents has changed. In particular, as shown by the examples in Figure 3 there have been an increasing number of accidents and of near-misses caused by the pilots' interaction with automation [14]–[16].

Mitigation for human-automation interaction errors consists primarily of defining concepts of operation in which the roles and responsibilities of the pilots and of the automation are clearly laid out. The problem here is that our current generation of cockpit systems are now so complex that, in emergency situations, misunderstandings about the current state and the most important monitoring and control tasks cause life-threatening problems. As a result, there is active research on using models of human-machine interaction in conjunction with model-checking in order to find these aviation cockpit (and control tower) errors at the time of automation design [17]–[19]. Accessing the interactions between pilots and plans is a difficult task. Collecting enough data about pilots' behavior, repeated for an adequate sample of flight conditions, can take months to years or even decades. Worse, if some conclusion of that study is subsequently challenged, then it is very difficult to reproduce the conditions that led to the original conclusions.

In response to this issue, researchers have built models of behavior of humans performing in those safety critical systems. Using those models, analysts can quickly exercise more scenarios in a reproducible manner. For example, researchers at Georgia Tech [8]–[11], [20], [21] have developed the WMC (Work Models that Compute) framework. WMC's CDA (continuous descent approach) simulation models the physics of flight and the atomic actions of the pilots and the automation, in the context of prioritization schemes.

CDA can understood by contrasting it with another landing tactic, the *standard descent*. In a standard descent, an aircraft must descend via several steps, requesting a new clearance at every step. As a consequence, flight times are longer and more fuel is burned. Also, at lower steps, the aircraft is effectively closer to the city itself, emitting lots of noise as the aircraft passes by. As the aircraft reroutes and encircles an airport before a runway is clear to land on, these wait times equate

CDA = Continuous Descent Approach
CPU = Central Processing Unit
GALE = Geometric Active LEarning
HTM = Maximum Human Task load
MOO = Multi-objective optimization
WMC = Working Models that Compute

Fig. 2: Acronyms in this paper.

.

In the 2009 Air France 447 crash [22], the autopilot disengaged when it lost airspeed data from the pitot tubes. A copilot lost situational awareness, and believed he was in 'take-off, go-around' mode, in which the appropriate response is to climb. However, at 38,000 feet, the rate of climb that the copilot was commanding was unsustainable; Air France 447 stalled. During the one sequence in which the copilot responded appropriately and dropped the plane's nose, the plane gained enough airspeed that the autopilot started working again and began issuing a stall warning; this effectively punished the copilot for reacting appropriately. Two other, more experienced pilots on the flight deck scanned the many screens and gauges available to them, but failed to notice the fact that the copilot was commanding a climb. One of the pieces of data that might have helped the pilots understand their situation was the plane's angle-of-attack, but this information was available only to the automation.

Asiana Flight 214 crashed in June of 2013 at the San Francisco International Airport (SFO). The final report suggests that this crash may also have been partly due to confusion about the division of labor between the pilots and the automation [23]. Ground automation at SFO (the precision instrument landing system) was not working, and the pilots of Asiana 214 were asked to fly a manual approach. The plane's tail struck the seawall at the end of the runway; the plane was too slow and too low for landing. Both pilots reported that, until the last few seconds of the crash, they believed the autothrottle was controlling the plane's speed. The autothrottle was in an 'armed' position, but was not on. The pilot flying's flight director (another system for automation in the cockpit) was *deactivated* and the instructor pilot's flight director was *activated*; had the systems been in the same state, the autothrottle would have 'woken up'. The flight crew failed to monitor airspeed, and the investigators concluded that fatigue and over-reliance on automation were primary contributors to the accident.

Fig. 3: Two examples of accidents in which complex cockpits played a role.

to more noise for the city. Additionally, it is harder to fly at lower altitudes due to changes in the atmosphere and wind environments.

By contrast a *continuous descent approach* is a continuous non-stepped descent in which only one request for landing is needed. This simplifies the communication overhead between radio towers and pilots, and avoids extended duration at low altitude. As a result, a continuous descent can more accurately approach the runway, less fuel is burned, and less noise is emitted into the city. Figure 4 illustrates the difference between CDA and traditional landing methods.

III. Learning From Models

For analyzing complex models like CDA, we prefer GALE to traditional optimizers since those optimizers require certain simplification assumptions. For example, if models are simple continuous equations, then they could be readily explored with gradient descent methods such as the Quasi-Newton method (perhaps using the BFGS update rule recommended by Sims [24]). However, all such gradient descent methods assume that the model being explored is essentially continuous. Models like CDA are not continuous since their internal state space is divided into one combination of each branch of each *if statement* in the code [25].

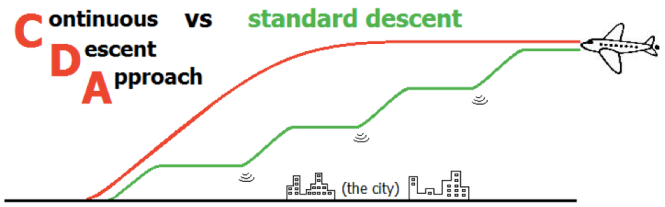

Fig. 4: The red line shows a Continuous Descent Approach (CDA). The green stepped line represents the traditional approach, which is closer to the city, so the city hears more noise emitted from the aircraft.

.

A better approach for exploring complex models is a multi-objective evolutionary algorithm (MOEA). There are many such optimizers including GALE and more traditional tools such as NSGA-II [4]. MOEAs assume that the model is a function that converts *decisions* "d" into *objective* scores "o"; i.e.

$$o = model(d)$$

In this framework, each pair (d, o) is an *individual* within a *population*. MOEAs try to find a range of good inputs by progressively improving the population using the *generational* approach of Figure 5. Note that MOEAs make no assumptions about model continuity (e.g. unlike the Quasi-Newton methods, they do not assume that models have local smooth gradients). They can also explore trade-offs between goals (see the domination predicate discussed in Figure 5). As discussed below, MOEAs have problems with *brittleness* and *CPU*.

An evolutionary multi-objective optimization algorithm (MOEA) requires at least two operators: *cull* and *perturb*: MOEAs generate an initial population by randomly selecting decisions then *culling* the individuals with the lower objective scores. A new population P_n is generated by *perturbing* the decisions of the surviving individuals (e.g. via random mutation or grafting together parts of the decisions of different individuals). MOEA's halt when P_n scores no better than prior generations $P_{m<n}$.

One way to implement the culling (step 2) is via *domination*; i.e. remove one example if it can be shown that it is worse than (a.k.a. "is dominated by") some other examples. Two forms of domination are *binary* and *continuous* domination. In *binary domination*, one individual x dominates y if all of x's objectives are never worse than the objectives in y but at least one objective in solution x is better than its counterpart in y; i.e.

$$\{\forall o_j \in objectives \mid \neg(o_{j,x} \prec o_{j,y})\}$$
$$\{\exists o_j \in objectives \mid o_{j,x} \succ y_{j,y}\}$$

where $(\prec, \succ)$ tests if an objective score in one individual is (worse,better) than in the other individual.

An alternate culling method is the *continuous domination* predicate [26] that favors y over x if x "losses" least:

$$\begin{aligned} worse(x,y) &= loss(x,y) > loss(y,x) \\ loss(x,y) &= \textstyle\sum_j^n -e^{\Delta(j,x,y,n)}/n \\ \Delta(j,x,y,n) &= w_j(o_{j,x} - o_{j,y})/n \end{aligned} \quad (1)$$

where "n" is the number of objectives and $w_j \in \{-1, 1\}$ depending on whether we seek to maximize goal x_j.

Fig. 5: Multi-objective evolutionary algorithms.

A. Brittleness

Ideally, any insight we glean from a model is not "brittle"; i.e. it is a conclusion that is robust in the face of minor changes to model inputs. Unfortunately, experts in MOEA reasoning caution that many MOEAs generate "brittle" decisions.

According to Harman [27], understanding the neighborhood around individual solutions is an open and pressing issue:

> "It may be better to locate an area of the search space that is rich in fit solutions, rather than identifying an even better solution that is surrounded by a set of far less fit solutions." [28].

He argues that many software model problems are *over-constrained*; i.e. no precise solution over all variables is achievable. Such over-constrained problems are usually explored using heuristic search methods such as the MOEA of Figure 5. The results of such partial heuristic search may be "brittle"; i.e., small changes to the search results may dramatically alter the effectiveness of the solution [28]. One way to check for brittleness is to use *neighborhood perturbation*:

1) *Cluster* a population into local neighborhoods;
2) Build a new population by *perturb*ing the decisions in each neighborhood;
3) Halt if objectives do not change after perturbation;
4) Else, go to step 1.

One reason we endorse the GALE algorithm for reaching conclusions from complex cognitive models is that it directly implements neighborhood perturbation.

B. Problems with CPU

CDA is a complex model with many input parameters. The input parameter space for such models tends to grow very large so there is a pressing and urgent need for efficient modeling techniques.

The primary design criteria for standard MOEAs is "ability to explore complex trade-offs" and not runtime speed. While the internal details of standard MOEAS may be very different (see Figure 6); most of them share one key characteristic: they evaluate $O(2N)$ examples (twice the population size because they generate offspring which are perturbations of their parent examples) for each generation. One reason we advocate GALE is that it replaces $O(2N)$ with a much faster $O(log_2 N)$ technique (see below).

GALE combines (a) the neighborhood perturbation (described above) with (b) the MOEA algorithm of Figure 5. The algorithm reflects over a *population* of points, each of which contains *decisions* (some inputs to a model). It then searches for the input decisions that lead to best outcomes. For example:

- if we adjust the inputs to the model for (say) high tailwind conditions...
- ... then GALE can report the best monitoring policies for the cockpit instrumentation.

Figure 7 lists the procedure by which GALE clusters the data into neighborhoods, then perturbs each neighborhood. In terms of monitoring for brittleness, the key point of GALE is that this process continues until the perturbations stop

- NSGA-II [4] uses a non-dominating sort procedure to divide the solutions into *bands* where $band_i$ dominates all of the solutions in $band_{j>i}$ (and NSGA-II favors the least-crowded solutions in the better bands).
- *SPEA2*: favors solutions that dominate the most number of other solutions that are not nearby (to break ties, it uses density sampling) [29];
- *IBEA*: uses continuous dominance to find the solutions that dominate all others [26];
- In *Particle swarm optimization* (PSO), a *particle*'s velocity is 'pulled' towards the individual and the community's best current solution [30];
- The *many-objective optimizers* are designed for very large numbers of objectives [31];
- Multi-objective *differential evolution* (DE): members of the frontier compete (and are possibly replaced) by candidates generated by extrapolation from any three other members of the frontier [32], [33];
- The *decomposition methods* that *first* divide the space of candidate solutions into numerous small regions; then *second* run a relatively simple optimizer in each region [34], [35].

Fig. 6: Some sample MOEAs. Note that this list is not exhaustive since this is a very active area of research.

GALE initially builds a population of points by selecting decisions at random. It then *clusters* those decisions into neighborhoods as follows:

1) Find two distant points in that population; call them the *east* and *west* poles.
2) Draw an axis of length c between the poles.
3) Let each point be at distance a, b to the *east, west* poles. Using the cosine rule, project each point onto the axis at $x = (a^2 + c^2 - b^2)/(2c)$.
4) Using the median x value, divide the population.
5) For each half that is larger than $\sqrt{N}$ of the original population, go to step 1.

Note that the above requires a distance measure between sets of decisions: GALE uses the standard case-based reasoning measure defined by Aha et al. [36]. Note also that GALE implements step 1 via the FASTMAP [37] linear-time heuristic:

- Pick any point at random;
- Let *east* be the point furthest from that point;
- Let *west* be the point furthest from *east*.

These final sub-divisions found by this process are the *neighborhoods* that GALE will *perturb* as follows:

- Find the objective scores of the *east, west* poles in each neighborhood.
- Using the continuous domination predicate of Figure 5, find the *better* pole.
- Perturb all points in that neighborhood by pushing them towards the better pole, by a distance $c/2$ (recall that c is the distance between the poles).
- Let generation $i+1$ be the combination of all pushed points from all neighborhoods.

From a formal perspective, GALE is an active learner [38] that builds a piecewise linear approximation to the Pareto frontier [39]. For each piece, it then pushes the neighborhood up the local gradient. This approximation is built in the reduced dimensional space found by the FASTMAP Nyström approximation to the first component of PCA [40].

Fig. 7: Inside GALE

having any new effect (i.e. they stop generating better objective scores). That is, all GALE solutions are guaranteed not to be brittle.

Also, in terms of reducing runtime, the key feature of GALE is that, unlike traditional MOEAs such as NSGA-II [4], GALE does not evaluate its entire population. Instead, as it recursively clusters the data in two (using steps 1,2,3,4,5 in Figure 7), GALE only computes the objective scores for the two most distant points in each division. This means that this binary division of the data terminates after just $log_2(N)$ comparisons of evaluated individuals. This is much less than the $2N$ evaluations required by traditional methods like NSGA-II.

Note that the above is a very brief description on GALE. For a full description including algorithms, download sites, and results from dozens of models, see [2], [3].

IV. A Case Study: CDA and GALE

This case study was designed after reflecting on the following. Sometimes, when monitoring cockpit instruments, a pilot is unable to complete all of their required tasks. Such lack-of-attention can have adverse consequences on aviation safety. It can occur for many reasons, including (1) fatigue; or (2) unexpected or stressful situations such as:

- Unexpected flight conditions such as increased tail winds or unscheduled rerouting;
- Emergency situations (e.g. a threatened near miss with another aircraft);
- Training situations in which one pilot must monitor a plane at the same time as watching over another, less experienced, pilot.

A. Goals

The goals of this study are:

1) To understand the safety implications of lack-of-attention;
2) To learn what mitigations exist (if any) for reducing safety problems associated with lack-of-attention.

We use interrupted, forgotten, and delayed tasks within the CDA model as an incomplete proxy for a safety metric. Our possible mitigations are restricted to the input of our CDA problem space.

B. Methods

1) Apparatus: This study uses the CDA model as described herein. CDA is a model of pilot interactions: with each other and also with the navigation systems critical to safe flight. CDA employs a continuous descent approach to a runway. As aforementioned, a continuous descent is an alternative to the standard approach to a runway. A continuous descent approach is arguably much more efficient in terms of a) fuel economy and cost, b) noise, and c) flight duration.

CDA is packaged within the WMC (Work Models that Compute) suite. WMC has the capability of modeling the way humans select which task to do next. In the CDA model, pilots have a handful of requirements and tasks that they must satisfy, and each requirement and task has a given level of priority that affects the order in which it is satisfied. WMC also takes into account that humans have a "maximum human task load" variable which describes how much they can handle. If the task load is too large, then some of the workload will not be completed in time, or worse yet, might go forgotten entirely. We use the metrics that describe these problems as the output objectives for GALE, i.e., the dependent variables which are detailed in the subsection just below.

Different strategies of pilot interaction can help alleviate the problems of large task loads. These different strategies are employed in CDA through its inputs, i.e. the independent variables as described in the next subsection. By studying the different strategies in terms of the output objective scores (the dependent variables), it is possible to understand the safety implications of different strategies and to learn what mitigations might exist (and can be exploited for safety gain).

2) Independent Variables: A CDA "problem instance" defined within the WMC framework consists of four decisions and five objectives. The CDA implementation itself within WMC contains many other inputs (for example, the flight path and aircraft type are fixed) but for the purposes of this study, they are held constant.

CDA's four decision variables are:

HTM: maximum human task load. This value describes how many tasks (where a task is an atomic action) can be maintained in a mental to-do list by a person. Tasks in the model are assigned a duration and a priority. For a thorough description of this variable, please see [8]. When the number of necessary tasks exceeds the number of tasks that the person can maintain, there can be incurred delays, errors, or the possibility of the task being forgotten and lost.

FA: function allocation. This variable refers mainly to the relative authority between the human pilot and the avionics, and is discussed in more detail to follow.

CCM: contextual control mode of pilots. These describe the pilots' ability to apply patterns of activity in response to the demands and resources in the environment, and are described in detail below.;

SC: the air environment scenario. WMC's CDA model includes four different arrival and approach scenarios.

The four arrival and approach *scenarios* (SC) implemented within the CDA model are:

Nominal: (ideal) arrival and approach.

Late Descent: controller delays the initial descent.

Unpredicted rerouting: pilots directed to an unexpected waypoint.

Tailwind: wind pushes plane from ideal trajectory.

The *function allocation* (FA) defines the different ways the pilots can configure the autoflight controls. We list the different possible modes within the CDA below, interested readers are referred to [20] for more details.

Highly Automated: The computer processes most of the flight instructions directly; the pilot only confirms the clearances.

Mostly Automated: The pilot processes the instructions and programs the autoflight sytem, but then the autoflight

system controls the flight path automatically. This mimics current "LNAV" and "VNAV" flight operations.

Mixed-Automated: The pilot processes the instructions and programs the computer to handle the lateral flight path. The flight crew directly flies the vertical profile, including the altitude, vertical speed, and airspeed.

CDA also knows of three different pilot *contextual control modes* (CCM). These are based on Hollnagel's work on representative patterns of activity [41]. For more information on how these CCM's are implemented as sets of actions within WMC, please refer to [21].

Opportunistic: Pilots monitor and perform tasks related to only the most critical functions.

Tactical: Pilots cycle through most of the available monitoring tasks, and double check some of the computer's tasks.

Strategic: Pilots cycle through all of the available monitoring tasks, and try to anticipate future tasks.

3) Dependent Variables: CDA's five objectives keep track of how many tasks were delayed, interrupted or forgotten entirely (which impacts the relative safety of the flight itself). We summarize these metrics below; the interested reader should see [8] for full details. Better pilot organizational structures can be found by exploring different inputs of CDA to optimize these goals so that safety is improved:

Num Forgotten Actions: tasks forgotten by the pilot. When the number of tasks expected of the pilot exceed the HTM, tasks with the lowest priority are 'forgotten'.

Num Delayed Actions: number of delayed actions. Tasks have a scheduled time and a duration. When a higher priority task causes another task to begin later than its scheduled time, it is 'delayed'.

Num Interrupted Actions: interrupted actions. A higher priority task can cause a lower priority task to be interrupted.

Interrupted Time: time spent on interruptions.

Delayed Time: total time of all of the delays.

In CDA, the HTM (maximum task load) variable controls how many tasks a pilot can either perform or hold in working memory. In all the following, CDA was run for varying and decreasing values of HTM. For each HTM value, we collect:

- *The baseline objective scores* seen in CDA. Recall from the above that those baseline values relate to number of forgotten actions; delayed actions; interrupted actions; total interrupted time; etc.
- *The treated objective scores* seen in CDA. These treatments are learned by GALE and represent the best case actions that could be performed by pilots to mitigate against the lack-of-attention problem.

When analyzing CDA, GALE was run using parameters found to work best on several other models [2]:

- GALE uses a population of size 100;
- GALE must terminate after a maximum number of 20 generations;
- GALE may terminate if no improvement is seen in any objective in the last three generations;

To control for random effects during optimization, all scores are the mean values of 20 repeated runs of *baseline* or the model *treated* with GALE's conclusions.

4) Data Analysis: Originally, we only planned one experiment, called *Experiment #1*, to compare *baseline* and *treated* by letting GALE select any inputs across the full range of all CDA input values.

That first experiment found a curious threshold effect: underneath a certain HTM point, all the best decisions concerned a particular contextual control mode. As described above, in *Opportunistic* mode, pilots monitored and performed tasks related to only the most critical functions in the cockpit. That is, in this mode, pilots executed only the most essential monitoring actions according to the CDA model (e.g. monitoring altitude and monitoring descent airspeed), and focused primarily on adjusting the lateral profile. For full details, see [8], pages 110-135.

To understand the impact of this *Opportunistic* mode, an *Experiment #2* was conducted, exactly like Experiment #1, but with the *Opportunistic* mode disabled.

5) Sanity Checks: We define two sanity checks on our results:

Sanity check #1: The clear pre-experimental intuition is that, as we *decrease* HTM (maximum human task load), we should see *increasing* adverse flight operations. If this observation was not seen in the results, the entire investigation should be doubted.

Sanity check #2: Using CDA and GALE, we can find mitigations that reduce the effects of decreasing HTM. If this were otherwise, that would suggest that MOEAs like GALE are not useful here.

V. Results

A. Experiment #1: Results

Figure 8 and Figure 9 show the results of these two experiments. As an aid to help visualization, bar graphs are added to each column (and the bar with the largest, smallest value shows the max,min values in the column). In those figures, GALE's decisions are shown at top. Beneath that, we show two sets of objective scores:

- *The baseline runs* which are the average objective scores seen without using GALE (just from randomly selecting input decisions).
- *The treated runs* which are the average final objective scores seen after 20 runs of GALE.

For the tables of objective results:

- The controlled value (HTM, maximum human task load) is shown on the left hand side of the table.
- The values in the other columns are all counts per simulated minute.

When reading these results, it is insightful to look for *saturation*, *trends*, *absences* and *cliffs*.

1) Saturation: Values *saturate* when they are driven towards their theoretical maximum. In the case of the CDA objectives, any *Time* objective that occurs at 60 seconds per minute is *saturated*. Such saturation can be observed in the *avg Interrupted Time* and the *avg Delayed Time* values of 60 at $HTM = 1$ in Figure 8b and Figure 9b.

In terms of a pilot maintaining safe operations, saturation is highly undesirable. At saturation, a pilot is permanently

	Level of Autonomy			Scenario				Contextual Control Mode		
HTM	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	88%	12%	0%	31%	31%	24%	14%	100%	0%	0%
2	14%	86%	0%	16%	12%	28%	44%	96%	4%	0%
3	12%	84%	4%	26%	12%	23%	39%	72%	26%	2%
4	33%	57%	10%	33%	12%	31%	24%	14%	84%	2%
5	41%	57%	2%	39%	16%	26%	18%	8%	87%	5%
6	21%	79%	0%	31%	15%	44%	10%	2%	98%	0%
7	46%	52%	2%	48%	14%	21%	16%	4%	91%	5%
8	32%	68%	0%	30%	20%	38%	13%	4%	91%	5%

HIGH = Highly Automated.
MOST = Mostly Automated.
MIX = Mixed Auto/Manual

NORM = Nominal Descent
LATE = Late Descent Scenario
ROUT = Unexpected Rerouting
WIND = Unexpected Tailwind

OPP = Opportunistic
TAC = Tactical
STR = Strategical

Figure 8a: Exp #1. Decisions found by GALE, all contextual control modes enabled.

	HTM = Max Human Taskload	nFA= num Forgotten Actions (per minute)	nDA = num Delayed Actions (per minute)	nIA = num Interupted Actions (per minute)	AIT = avg Interupted Time (secs per minute)	aDT = avg Delayed Time (secs per minute)
baseline	1	1087	34	9	60	21
(no	2	752	11	5	13	7
optimizing)	3	466	5	3	3	3
	4	270	2	2	1	2
	5	151	1	2	0	1
	6	99	0	1	0	1
	7	57	0	1	0	1
	8	18	0	0	0	0
GALE	1	54	2	0	8	0
(optimizing)	2	39	0	0	1	1
	3	46	0	0	0	1
	4	140	1	1	1	1
	5	108	1	1	0	1
	6	61	0	1	0	1
	7	21	0	1	0	1
	8	1	0	0	0	0

Figure 8b: Exp #1. Objectives obtained, all contextual control modes enabled.

Fig. 8: Experiment #1: All contextual control modes enabled. Forgotten, delayed, and interrupted actions are reported by the simulation at each 0.05s time step. E.g. a monitoring action can be 'forgotten' 20 times each second.

interrupted for all tasks so everything gets delayed. Note that this saturation result satisfies one of our *sanity checks* (that less HTM leads to more problems).

The good news is that saturation can be avoided. In Figure 8b, we see that the simulated pilots following GALE's advice never reach saturation at $HTM = 1$. GALE advises the simulated pilots to restrict themselves to only the most important tasks (in CDA's model, this means operating in *opportunistic* mode) and to allow the automation to handle all or most of the tasks that the automation can handle. Note that this means that our experiment satisfies another *sanity check* (that GALE can learn mitigations to the low HTM problems).

2) Trends: *Trends* are values that change smoothly with changes to HTM. For example, the *num Forgotten Acts* per minute is low in all results until HTM falls below four (after which time it can spike to alarmingly large values).

With one exception, this trend holds for all objectives—which is to say that airplane safety is critically dependent on this HTM value.

The exceptions to this trend are the GALE results of Figure 8b. In those rows, GALE could learn mitigations that compensate for pilots struggling to control. Those mitigations are shown in Figure 8a.

3) Absence: Several columns in Figure 8a and Figure 9a contain nearly all zero values. That is they are mostly *absent* from the recommendations made by GALE.

Sometimes, these absences are not informative. For example, in Figure 9a, the absent values in *OPP* are the result of that experiment (this mode was disabled for that experiment). But other absent columns are more interesting:

- A *MIX*ed level of autonomy was rarely useful, suggesting that the simulated pilots should very rarely program the computer to handle only some of the airplane instructions.
- Similarly, the *STR* strategic cognitive control mode was also rarely used. From this result, we say that, in this model, pilots should avoid cycling through all of the available monitoring tasks while trying to anticipate future tasks.
- Other absent columns can be seen in the scenarios GALE found it could handle. In Figure 9b, GALE found that when opportunistic mode was disabled, it rarely could handle the *LATE* approach or high tail *WIND* situations. On the other hand, when opportunistic model was enabled, GALE's recommendations to the simulated pilot could handle all the *Scenario*s (see all the non-zero numbers in the *Scenario* columns of Figure 8b).

4) Cliffs: Cliffs are values that change sharply between one HTM value and the next; there are two large cliffs in Figure 8.

The first cliff relates to *Level of Autonomy*. At $HTM = 1$, GALE nearly always selected for a *HIGH* level of autonomy. However, as soon as pilots can do or hold in memory two things at once (at $HTM > 1$), that recommendation no longer holds. In fact, for all levels of $HTM > 1$, GALE usually prefers for the pilot to process flight data and program the computer (i.e. to use the *MOST* approach). Another way to say this is that, when their attention is failing, our simulated human pilots should give more tasks to the machines. However, at any other time, it is better to guide, and not be guided by, the automatic systems.

The other cliff in these results is seen in the right-hand-side columns of Figure 8a. In those results, it can be seen that for $1 \leq HTM \leq 3$ the *OPP* (opportunistic) cognitive control mode is most often selected by GALE. However, above that point (for $HTM > 3$), it is rarely selected. This cliff is a large enough effect in a critical range of the model to deserve special attention. Accordingly, Experiment #2 (discussed below) explores the the relative merits of opportunistic control versus other modes.

B. Experiment #2: Disabling Opportunistic Mode

As shown in Figure 8a, at low HTM levels of $HTM < 4$, most of GALE's recommended actions use the opportunistic cognitive control mode (where pilots monitor and perform tasks related to only the most critical functions). To see if this was some quirk of the simulation, or an important effect, we repeated the above experiment with this opportunistic mode disabled.

The results are shown in Figure 9. In those results, the following aspects are noteworthy:

- Comparing the *baseline* and *GALE* distributions of Figure 9b, we see that the GALE treatment barely changes the baseline distributions. That is, if GALE cannot use the opportunistic control mode, then it cannot mitigate for low HTM values.
- Comparing the *Scenario* results in Figure 9a to Figure 8a, we see that when we cannot use opportunistic mode there are more absent columns in *LATE* and *WIND*. That is, if GALE cannot use opportunistic control, then it cannot find another mitigation for late arrival or high wind conditions.

From these results, we say that opportunistic mode is an essential tool for combating the problems associated with low HTM in the CDA model.

VI. Discussion

In principle, all the above conclusions could have been reached using an MOEA like NSGA-II. However, in practice, that would have been impractically slow. To understand why, we must review the systems-level tasks associated with conducting this kind of study. Note that a discussion of these systems-level tasks rarely occurs in research publications. Researchers present only their final results and do not mention the work required before those results can be collected. In the case of this study, that work was quite extensive. *Commissioning* this CDA model took several months as one of us (Krall) worked inside the NASA firewalls to port the CDA model to the NASA servers. In that process, CDA was run many times to "iron out the bugs". Often it was necessary to trace through the evaluations to determine what was going astray. During this period, we were grateful that GALE was only making $O(log(N))$ evaluations per generation since the $O(2N)$ evaluations used by standard optimizers would have led to an overwhelming amount of data.

Also note that after commissioning the model came *generating conclusions*. This required 20 repeats of all models for baselines and with GALE, repeated for HTM set from one to eight, then repeated twice (for Experiment #1 and #2). With GALE, those runs took 83 hours and with NSGA-II, those runs would have taken much longer. Based on some samples we made of NSGA-II performing parts of Experiment #1, we estimate that if NSGA-II was used for the above experiments, then that would have taken 6 months.

As to the external validity of this work, all the conclusions made here came from two tools: the CDA model and GALE. If these tools are somehow distorted or biased then our conclusions would be distorted or biased in the same manner. That said, there is enough prior work on CDA and GALE to make the case that it useful to study the CDA model with the GALE optimizer. We note that these tools are the products of years of research and much analysis and testing [1]–[3], [8]–[12], [20], [21]. CDA is one of the largest and most studied models of pilot cognition currently available. Given the resources spent on its construction, it seems prudent and timely to learn what we can from that model.

VII. Conclusion

A common, and naive, assumption made by researchers who have not conducted model-based experiments is this: once the model is built, then inference is easy.

We have shown in this paper that, for large and complex models, this naive assumption may not hold. In fact, it is critically important to consider *how* that inference is conducted. This paper endorses the use of modeling for complex studies,

	Level of Autonomy			Scenario				Contextual Control Mode		
HTM	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	66%	32%	2%	35%	21%	17%	26%	x	94%	6%
2	87%	13%	0%	48%	0%	21%	31%	x	94%	6%
3	43%	55%	2%	78%	0%	20%	2%	x	96%	4%
4	40%	60%	0%	73%	0%	21%	6%	x	100%	0%
5	67%	33%	0%	81%	2%	18%	0%	x	95%	5%
6	27%	73%	0%	80%	2%	18%	0%	x	100%	0%
7	38%	60%	2%	83%	2%	15%	0%	x	98%	2%
8	43%	57%	0%	81%	0%	19%	0%	x	98%	2%

HIGH = Highly Automated.
MOST = Mostly Automated.
MIX = Mixed Auto/Manual

NORM = Nominal Descent
LATE = Late Descent Scenario
ROUT = Unexpected Rerouting
WIND = Unexpected Tailwind

OPP = Opportunistic
TAC = Tactical
STR = Strategical

Figure 9a: Exp #2. Decisions found by GALE, opportunistic mode disabled.

		HTM = Max Human Taskload	nFA= num Forgotten Actions (per minute)	nDA = num Delayed Actions (per minute)	nIA = num Interupted Actions (per minute)	AIT = avg Interupted Time (secs per minute)	aDT = avg Delayed Time (secs per minute)
baseline (no optimizing)	1	1333	44	11	60	25
	2	958	14	6	15	8
	3	562	6	4	4	4
	4	295	2	2	1	2
	5	151	1	2	0	1
	6	100	0	1	0	1
	7	60	0	1	0	1
	8	19	0	0	0	0
GALE (optimizing)	1	1389	38	11	60	27
	2	684	10	5	8	7
	3	394	4	3	2	3
	4	189	2	2	1	1
	5	102	1	1	0	1
	6	56	0	1	0	1
	7	19	0	1	0	1
	8	1	0	0	0	0

Figure 9b: Exp #2. Objectives obtained, opportunistic mode disabled.

Fig. 9: Experiment #2: Opportunistic mode disabled. See Figure 8 for a description of how the tasks are counted.

but it also addresses a few issues that must also be handled whenever learners are used in conjunction with models.

Firstly, there is a need to develop and commission the model for integration with the learner tools. This can be a time-intensive task and moreover, additional modifications can be cause for restarting the actual experimentation which follows thereafter.

Secondly, the learners themselves are complex computational intelligence software tools which have been the subject of decades of research. The selection and deployment of just a single tool can become a complex decision process. Moreover, high model runtimes can restrain the space of usable learning tools by a vast amount as much of the research on those learners has focused on optimizing very small models.

In this paper, GALE was used because it can optimize very large models. This sort of enabling technology is made possible because GALE does not need to run the models as often as other learning tools (specifically, GALE performs $O(log(N))$ evaluations while standard methods explore a space of $O(2N)$ options). GALE can quickly generate conclusions from complex models. For example, in the case study explored here we offer the following answers to the research questions posed in the introduction.

A. *RQ1: Safety and Low HTM*

We say that unsafe operation occurs if pilots cannot complete their assigned tasks. For little to no cognitive limitations (HTM is 4 or higher), there is very little effect on taskload completion times. The reason for this is simple: if the pilot can perform all tasks as they come, then there should not be any backlog of delayed tasks. For lower levels of HTM, there were many delays and interruptions noted for our simulated

pilots. Hence, we conclude that low HTM levels are especially dangerous within this model.

B. RQ2: Impact of Automation

For nearly all levels of HTM, it was sufficient to rely on a *MOST* level of autonomy where the pilot is in charge of processing input and programming the cockpit computers. However, in extreme situations ($HTM = 1$), that recommendation is not supported. For such very low levels of HTM, our simulated pilots should switch to *HIGH* levels of automation to ensure aircraft safety.

This recommendation intuitively makes sense if we assume that the automation works as the pilot intends. Automation failures and automation surprise are beyond the scope of our study; we note that the creators of WMC have recently published a study with a version of their model tuned to study automation surprise [19].

C. RQ3: Pilot Monitoring Policies

As to appropriate cognitive control modes for watching over an aircraft, for higher levels of HTM, it was sufficient to step up to the tactical control mode (which allows the pilot to monitor more of the aircraft flight procedures). For lower levels of HTM, tactical flight operations proved to be too much for our simulated pilot to handle and opportunistic control mode was essential to mitigate against low HTM.

More validation should be done before applying the recommendations we make here for the CDA model to the real-world safety problems that have inspired both CDA and this study. However, our findings intuitively make sense. In both the AirFrance and Asiana accidents that we used in our motivation, a key finding was that the pilots became distracted by off-nominal behavior, and *failed to monitor* the most important flight state information (e.g. airspeed, altitude and attitude). When stressed, pilots are asked to do something very like the opportunistic mode as implemented within the CDA model—worry about the key monitoring and flight tasks first.

ACKNOWLEDGEMENTS

The work was funded by NSF grant CCF:1017330 and the Qatar/West Virginia University research grant NPRP 09-12-5-2-470. This research was partially conducted at NASA Ames Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

REFERENCES

[1] J. Krall, T. Menzies, and M. Davies, "Learning the task management space of an aircraft approach model," in *Proceedings of the 2014 AAAI Workshop*, ser. AAAI'14, 2014.

[2] J. Krall, "Faster evolutionary multi-objective optimization via GALE: the geometric active learner," Ph.D. dissertation, West Virginia University, 2014.

[3] J. Krall, T. Menzies, and M. Davies, "Geometric active learning for software engineering," *Under-Review, IEEE TSE*, 2014.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast elitist multi-objective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2000.

[5] A. Pingstone. (2007) Hornet moth dh87b g-adne arp.jpg. Released into the public domain. [Online]. Available: http://commons.wikimedia.org/wiki/File:Hornet_moth_dh87b_g-adne_arp.jpg

[6] A. Radecki. (2007) Butler-dc7-n6353c-071102-fox-tanker66-01-16.jpg. Photo released under the Creative Commons Attribution-Share Alike 3.0 Unported, 2.5 Generic, 2.0 Generic and 1.0 Generic license. [Online]. Available: http://commons.wikimedia.org/wiki/File:Butler-dc7-N6353C-071102-fox-tanker66-01-16.jpg

[7] Naddsy. (2007) Airbus A380 cockpit.jpg. Photo released under the Creative Commons Attribution 2.0 Generic license. [Online]. Available: http://www.flickr.com/photos/83823904@N00/64156219/

[8] S. Y. Kim, "Model-based metrics of human-automation function allocation in complex work environments," Ph.D. dissertation, Georgia Institute of Technology, 2011.

[9] A. R. Pritchett, H. C. Christmann, and M. S. Bigelow, "A simulation engine to predict multi-agent work in complex, dynamic, heterogeneous systems," in *IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, Miami Beach, FL, 2011.

[10] K. M. Feigh, M. C. Dorneich, and C. C. Hayes, "Toward a characterization of adaptive systems: A framework for researchers and system designers," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 54, no. 6, pp. 1008–1024, 2012.

[11] S. Y. Kim, A. R. Pritchett, and K. M. Feigh, "Measuring human-automation function allocation," *Journal of Cognitive Engineering and Decision Making*, vol. 8, no. 1, 2014.

[12] A. R. Pritchett, S. Y. Kim, and K. M. Feigh, "Modeling human-automation function allocation," *Journal of Cognitive Engineering and Decision Making*, vol. 8, no. 1, 2014.

[13] L. Coombs, *Control in the Sky: the Evolution and History of the Aircraft Cockpit*. Leo Cooper, Ltd., 2005.

[14] C. Billings, "Human-centered aircraft automation: A concept and guidelines," NASA, Tech. Rep. 103885, 1991.

[15] N. B. Sarter and D. D. Woods, ""from tool to agent": The evolution of (cockpit) automation and its impact on human-machine coordination," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 1995, pp. 79–83.

[16] N. Sarter, D. D. Woods, and C. Billings, "Automation surprises," in *Handbook of Human Factors & Ergonomics*, G. Salvendy, Ed. Wiley, 1997, pp. 543–501.

[17] M. Bolton, R. Siminiceanu, and E. Bass, "A systematic approach to model checking human-automation interaction using task-analytic models." *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 5, pp. 961–976, 2011.

[18] N. Rungta, G. Brat, W. Clancey, C. Linde, F. Raimondi, S. Chin, and M. Shafto, "Aviation safety: Modeling and analyzing complex interactions between humans and automated systems," in *International Conference on Application and Theory of Automation in Command and Control Systems (ATACCS)*, May 2013.

[19] G. Gelman, K. M. Feigh, and J. Rushby, "Example of a complementary use of model checking and human performance simulation," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 5, Oct. 2014.

[20] A. R. Pritchett, K. M. Feigh, S. Y. Kim., and S. K. Kannan, "Work models that compute to describe multiagent concepts of operation: Part 1." *Journal of Aerospace Information Systems*, vol. 11, no. 10, Oct. 2014.

[21] K. M. Feigh, A. R. Pritchett, S. Mamessier, and G. Gelman, "Generic agent models for simulations of concepts of operation: Part 2," *Journal of Aerospace Information Systems*, vol. 11, no. 10, Oct. 2014.

[22] "Final report on the accident on 1st June 2009 to the Airbus A330-203 registered F-GZCP operated by Air France flight AF 447 Rio de Janeiro - Paris," BEA, Tech. Rep., 2012.

[23] "Descent below visual glidepath and impact with seawall asiana airlines flight 214 boeing 777-200er, hl7742 san francisco, california july 6, 2013," National Transportation Safety Board, Tech. Rep., 2014.

[24] C. Sims, "Matlab optimization software," 1999.

[25] M. Davies, C. Pasareanu, and V. Raman, "Symbolic execution enhanced system testing," in *Verified Software: Theories, Tools, Experiments*. Springer Berlin Heidelberg, 2012, pp. 294–309.

[26] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*. Springer, 2004, pp. 832–842.

[27] M. Harman and B. Jones, "Search-based software engineering," *Journal of Information and Software Technology*, vol. 43, pp. 833–839, December 2001.

[28] M. Harman and J. Wegener, "Getting results from search-based approaches to software engineering," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 728–729.
[29] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation, and Control*. CIMNE, Barcelona, Spain, 2002, pp. 95–100.
[30] H. Pan, M. Zheng, and X. Han, "Particle swarm-simulated annealing fusion algorithm and its application in function optimization," in *International Conference on Computer Science and Software Engineering*, 2008, pp. 78–81.
[31] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, Aug 2014.
[32] R. Storn and K. Price, "Differential evolution– a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[33] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, Feb 2011.
[34] K. Deb, M. Mohan, and S. Mishra, "Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions." *Evolutionary Computation*, vol. 13, no. 4, pp. 501–525, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/ec/ec13.html#DebMM05
[35] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *Trans. Evol. Comp*, vol. 11, no. 6, pp. 712–731, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2007.892759
[36] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, January 1991.
[37] C. Faloutsos and K.-I. Lin, "FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, 1995, pp. 163–174.
[38] S. Dasgupta, "Analysis of a greedy active learning strategy," *in Neural Information Processing Systems 17:*, vol. 1, no. x, 2005.
[39] M. Zuluaga, A. Krause, G. Sergent, and M. Püschel, "Active learning for multi-objective optimization," in *International Conference on Machine Learning (ICML)*, 2013.
[40] J. C. Platt, "FastMap, MetricMap, and Landmark MDS are all Nyström algorithms," in *In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 261–268.
[41] E. Hollnagel, *Human Reliability Analysis: Context and Control*. London: Academic Press, 1993, pp. 159–202.

Joseph Krall (Ph.D., WVU) is a Postdoctoral Research Fellow funded by the National Science Foundation and is employed at LoadIQ, a high-tech start-up company in Reno, Nevada that researches and investigates cheaper energy solutions. His research relates to the application of intelligent machine learning and data mining algorithms to solve NP-Hard classification problems. Further research interests lie with multi-objective evolutionary algorithms, search based software engineering, games studies, game development, artificial intelligence, and data mining.

Tim Menzies (Ph.D., UNSW) is a Professor in CS at NcState and the author of over 200 referred publications. In terms of citations, he is one of the top 100 most most cited authors in software engineering (out of 54,000+ researchers, see http://goo.gl/vggy1). In his career, he has been a lead researcher on projects for NSF, NIJ, DoD, NASA, as well as joint research work with private companies. He teaches data mining and artificial intelligence and programming languages. Prof. Menzies is the co-founder of the PROMISE conference series (along with Jelber Sayyad) devoted to reproducible experiments in software engineering: see http://promisedata.googlecode.com. He is an associate editor of IEEE Transactions on Software Engineering. the Empirical Software Engineering Journal, and the Automated Software Engineering Journal. For more information, see his web site http://menzies.us or his vita at http://goo.gl/8eNhY or his list of publications at http://goo.gl/8KPKA .

Misty Davies (Ph.D. Stanford),is a Computer Research Engineer at NASA Ames Research Center, working within the Robust Software Engineering Technical Area. Her work focuses on predicting the behavior of complex, engineered systems early in design as a way to improve their safety, reliability, performance, and cost. Her approach combines nascent ideas within systems theory and within the mathematics of multi-scale physics modeling. For more information, see her web site http://ti.arc.nasa.gov/profile/mdavies or her list of publications at http://ti.arc.nasa.gov/profile/mdavies/papers.

RESPONSE TO REVIEWERS

Associate Editor's Comments

Here are my suggestions

1. Focus section 1 on GALE and its potential use in optimization of models and do not focus on WMC or CDA at all. Explain that GALE could be used to analyze cognitive models that in general are very complex and thus could use model-based analysis frameworks such as GALE to make analysis more tractable.

Yes. And following on from your suggestions, we moved much of section 4 into section1.

2. Delete section 2. I find it really is distracting. Pull the idea that there are too many analyses to run from section 2 into the new introduction but all the information about cockpit automation history and the 2 aviation examples are not necessary.

Here, I'd like to push back just a little on your editorial suggestions.

Certainly, this section can be extensively shortenned, and this draft does that.

But I want to point out that you work in this field and hence you do not need a strong motivation to read this paper.

Please consider another kind of reader, one who is only tangentially connected with this work. That reader would need some "gentiling" in order to read this article. For example, in terms of supplying that motivation to read the paper, I really like the phrase "a modern airplane is really a computer with wings" and those graphics of increasining fearsome cockpit complexity.

3. Delete the current Section 3 (it should go into a case study)

Agreed. We've wound an intro to CDA into the motivation section of the revised section 2.

4. The current section 4 should really be section 2. However the material at its start about model-based approaches should be moved to the new introduction. Get rid of the discussion of what you will explore later in the paper from section 2. Again no CDA, HTM, FA CCM and SC material should go there. From 4.1 delete all of the material about Kims thesis. The text in 4.1 could be very short. The text up until the problem here could just say simulations with many input parameters grow in factorial space and there is a need for efficient modeling techniques.

Yes. Done.

5. I cant really comments on Section 5 because this is your technology. However again take out all the CDA and WMC portions.

*Yes: and there is one para from there that works **much** better in the introduction anyway.*

6. Now the new section 3 should be the methods for your case study and the new section 4 should be the results of your case study.

Not quite, since I'm still trying to retain section2. But I think I've followed the spirit, if not the letter, of your suggestions

7. This new methods section should read like a methods section where you have apparatus, independent variables, dependent variables, data analysis and so on. In the way I read your paper, the WMC simulation was used to create data sets that you analyze using GALE. If I am incorrectly understanding this, of course then correct me. If I am right, you could explain WMC, the CDA model, the parameters it uses and the data set it creates in each run in an apparatus section. The dependent variables and independent variables sections would be those used by GALE. The data analysis part really is the part where you describe how you ran GALE (pop size of 100 and so on).

That "Methods" section is now divided as you said and contains five sub-headings: (1) apparatus, (2) independent variables; (3) depednent variables; (4) data analysis and (5) sanity checks.

8. In your new results section you would have a subsection for exp 1 and one for exp 2

Done

9. Section 6.6, 6.7 and Section 7 look like what should be in a discussion section

Done.

10. Section 8 can also be deleted

Ok

I totally get it if you find this really annoying but I hope you take these comments as a way to make your paper relevant to the journals readers. I think it could be shorter and focused on the story you are trying to tell.

No problem. Great to work with an editor that takes so much care and consideration!