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 that arise during approach to runway of a large commercial aircraft. In that situation, 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 such complex models with a combination of data miners (that find a small set of most critical examples) and of multi-objective optimizers (that focus on just 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, Evolutionary Algorithms, Active Learning

1 Introduction

As cockpit avionics grow more complex, it becomes increasingly important to analyze the interactions between pilots and flight software. To this end, researchers build models of pilots and aircraft and the flight environment. For example, researchers at Georgia Tech [1]–[5] have developed the WMC (Work Models that Compute) framework. One WMC model is CDA, which models a continuous descent approach for aircraft. CDA can simulate off-nominal scenarios (e.g. unexpected rerouting; tail wind gusts; increasing levels of delay). After simulating a touchdown, a report is generated which produces a track of the aircraft flight course (in response to pilot actions) as well as statistics on how well the pilots have operated the aircraft (e.g. number of delayed/ missed actions).

In theory, even complex models like CDA can be quickly analyzed by running them across many CPUs. In practice, there are two problems with that approach: (1) accessing such large CPU farms is still bounded by the available resources; (2) and even if those CPUs were accessible, then all that output may not be the best way to present complex results to an analyst.

To address these two problems, we propose the use of a novel optimization method, called GALE [6]–[8], for the analysis of complex models. GALE focuses on a small number of most informative examples. Consequently, GALE explores only a few dozen examples rather than the thousands (or more) used by traditional methods [6], [8]. This simplifies and improves our ability to reason about complex cognitive models like CDA.

Joseph Krall is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University; 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.

While the algorithms behind GALE have been presented previously [6]–[8], those prior reports were focused on runtimes and did not explore the analysis implications of GALE. The core contribution of this paper is a case study on how this GALE style of reasoning can assist in the analysis of cognitive modeling. To that end, this paper takes a large model of pilot cognition (CDA) 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 §4, (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. §2 motivates this paper with a review on how cockpits are becoming increasingly more complex. Such complex systems can be studied using model-based methods such as the CDA model described in §3. These models are intricate and take time to run. An effective search of the models utilizes automated tools such as the multi-objective objective evolutionary algorithms (MOEAs) discussed in §4. One such MOEA is our GALE tool discussed in §5 which, in §6 is applied to CDA. The final conclusion section of this paper offers answers to three research questions, in the context of the CDA model:

- RQ1: *Safety and Low HTM:* 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: *Impact of Automation:* What are the effects of changing how much pilots rely on automation?
- RQ3: *Pilot Monitoring Policies:* What are the effects of changing pilot policies for monitoring and overlooking flight procedures?

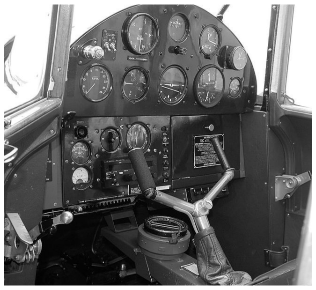


Figure 1a: Haviland Moth (1936) [9].

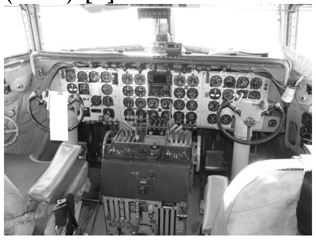


Figure 1b: DC7 (1953) [10].

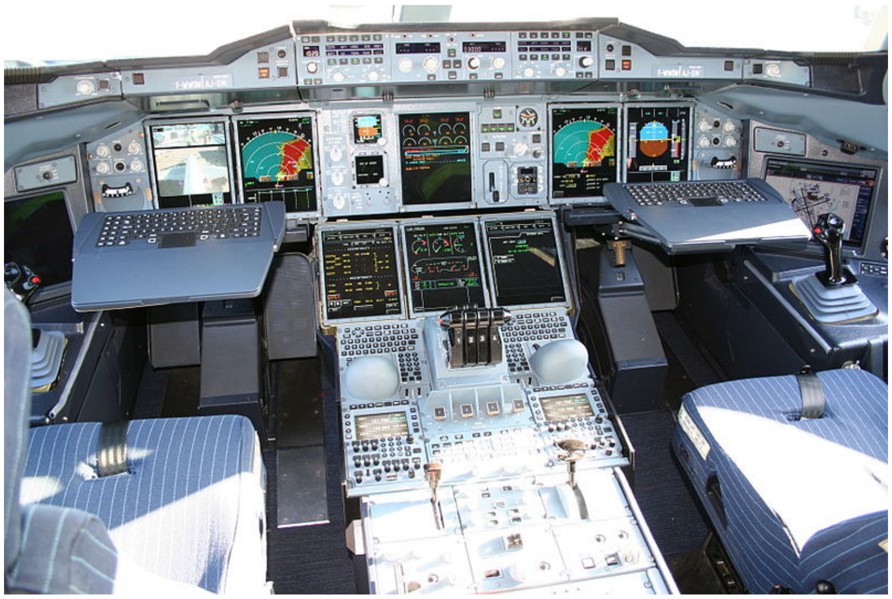


Figure 1c: Airbus 380 (2005) [11].

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

2 Motivation

This work is motivated by the growing complexity of cockpits. A cockpit in 1936 (see Figure 1a) 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. The pilot controls were placed according to mechanical convenience.

By 1953, as seen in Figure 1b, pilots had access to much more information. Civil transport aircraft now included redundant systems, including a copilot. Concepts of operation had to be developed to clearly define the roles and the responsibilities of the pilot and copilot.

By 2010, aircraft had become computers by wings; e.g. the Airbus 380 of Figure 1c. Autopilots allow the pilot to turn over much of the route planning and control to automation on the flight deck, and instead take a more passive monitoring role. This level of redundancy and automation, along with flight deck design that carefully considered a pilot's strengths and weaknesses is credited with increasing the safety of the airspace [12].

That said, while automation has certainly helped to reduce the number of aviation accidents, the kinds of errors that led to accidents has changed. In particular, there have been an increasing number of accidents and of near-misses caused by the pilots' interaction with automation [13]–[15]. 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.

Still, as recent aviation accidents can attest, there is still room for preventing error. Consider the examples of Air France 447 and Asiana 214.

In the 2009 Air France 447 crash [16], the autopilot disengaged when it lost airspeed data from the pitot tubes. A copilot lost situational awareness, and seemed to believe that 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 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 [17]. 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

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.

.

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.

In summary, 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. In the very near future, this situation could get much worse. Our current airspace is at an inflection point for increasing autonomy. NextGen air traffic control has a 'best-equipped best-served' model to encourage transport airlines to upgrade to equipment that will support new automation. The class of human-machine interaction errors can be expected to increase as a fraction of possibly preventable accidents. 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 [18]–[20]. While a promising line of research, model-checking has not yet scaled to the number of possible states in high-fidelity aviation models. In particular, aircraft trajectories, interactions between multiple levels of autonomous systems, and human cognitive systems are difficult to abstract. In addition, even high-fidelity models fall below necessary realism—currently, they can only be used to suggest avenues for study with human-in-the-loop experiments. It can be argued that, unless we discover some fundamental underlying symmetry, our models must become even more complex as they become more useful.

3 Model-Based Analysis

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, it is faster to exercise more scenarios in a reproducible manner. For example, Researchers at Georgia Tech [1]–[4], [21], [22] have developed the WMC (Work Models that Compute) framework. **This paper uses WMC's CDA (continuous descent approach) simulation, which models the physics of flight and the atomic actions of the pilots and the automation, in the context of prioritization schemes.**

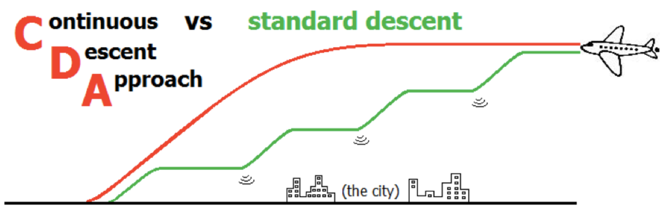


Fig. 3: The red line shows a Continuous Descent Approach (CDA). The green stepped line represents the traditional approach, which is closer to the city, so the city hears more noise emitted from the aircraft.

.

As shown in Figure 3, CDA can understood by contrasting it with another landing tactic called *standard descent*. In a standard descent, an aircraft must request clearance and descend via several steps, requesting 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 to more noise for the city. Additionally, it is harder to fly at lower altitudes due to changes in the atmosphere and wind environments.

Continuous descent approach offers a continuous non-stepped descent in which only a single request for landing is needed. This simplifies the communication overhead between radio towers and pilots, and also avoids extended duration at low altitude. In addition, a continuous descent can more accurately approach the runway, less fuel is burned, and less noise is emitted into the city.

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:

1) **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 [1]. 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.
2) **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.
3) **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.;

4) **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:

1) *Nominal* (ideal) arrival and approach.
2) *Late Descent*: controller delays the initial descent.
3) *Unpredicted rerouting*: pilots directed to an unexpected waypoint.
4) *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 [21] for more details.

1) *Highly Automated:* The computer processes most of the flight instructions directly; the flight control only confirms the clearances.
2) *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.
3) *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 [23]. For more information on how these CCM's are implemented as sets of actions within WMC, please refer to [22].

1) *Opportunistic*: Pilots monitor and perform tasks related to only the most critical functions.
2) *Tactical*: Pilots cycle through most of the available monitoring tasks, and double check some of the computer's tasks.
3) *Strategic*: Pilots cycle through all of the available monitoring tasks, and try to anticipate future tasks.

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 [1] 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:

1) *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'.
2) *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.*
3) Num Interrupted Actions*: interrupted actions. A higher priority task can cause a lower priority task to be interrupted.*
4) Interrupted Time*: time spent on interruptions.*
5) Delayed Time*: total time of all of the delays.*

4 Learning From Models

There are many advantages of a model-based approach to human factors. Traditional human-in-the-loop experimental case-studies are expensive and 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. For example, with CDA, a 20 minute descent path of an aircraft can be run in under 10 seconds. Due to this faster-than-real time speed, model-based methods can explore more options than slow-time real-world case studies.

The primary disadvantage of these models is that they are only an approximation to reality. The models can suggest underlying behavioral patterns, but these patterns must be confirmed in human-in-the-loop experiments; models can decrease time spent in these experiments, but cannot eliminate the experiments. Another disadvantage of models is, even at their reduced fidelity, they still represent a complex set of connections between input and output, and can take significant time and expertise to explore.

In practice, this emerges as the *feature interaction* problem [24]; i.e. some combination of two inputs has a different effect than inputting either one separately. Feature interaction means that it can be misleading to study models by holding $N-1$ inputs steady while altering the last input.

To address this problem, there are optimization algorithms that strive to explore the entire space of interactions within a model. With such tools, it is possible to define a *context* then find the best actions within that context. For example, later in this paper, we explore:

- What happens in the model when pilots are distracted from their normal cockpit monitoring tasks by some stress-inducing situation; e.g. unexpected rerouting or tail winds.
- What is the best we can expect from our modeled pilots in those different contexts.

For this task, we prefer GALE to traditional optimizers since those optimizers required 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 [25]. 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 [26].

An evolutionary multi-objective optimization algorithm requires at least two operators: *cull* and *perturb*:

1) Generate an initial population by randomly selecting decisions;
2) *Cull* the individuals with the lower objective scores.
3) Generate a new population P_n by *perturbing* the decisions of the surviving individuals (e.g. via random mutation or grafting together parts of the decisions of different individuals)
4) Halt if P_n no better than prior generations $P_{m<n}$.
5) Else, go to step 2.

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 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.

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

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 [27] that favors y over x if x "losses" least:

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

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

Fig. 4: Inside an MOEA.

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 [28]. MOEAs assume that 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*. For example, in the case of CDA, one (d, o) pair would record what happened when that model was run with certain values supplied for the HTM, FA, CCM and SC variables described in the previous section.

MOEAs try to find a range of good inputs by progressively improving the population using the *generational* approach of Figure 4. Note that MOEA 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 4). That said, these MOEA algorithms have problems with *CPU* and *brittleness*.

There any many other kinds of MOEA algorithms including the following (this list is not exhaustive; this is a very active area of research):

- NSGA-II [28] 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 [27];
- 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* 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].

4.1 Problems with CPU

Cognitive models are meant to run more quickly than human-in-the-loop experiments, but they still take time to run. The CDA model originally defined in Kim's Ph.D. thesis divides its input space 144 ways, each of which requires a separate simulation [1]. In our exploration, we further subdivide the maximum human taskload to evaluate 252 combinations. Worse yet, a detailed reading of Kim's thesis shows that these 144 input sets actually explore only one variant each for three of her inputs [1]. Other modes would need to be explored to handle:

- Unpredictable rerouting;
- Different tail wind conditions;
- Increasing levels of delay.

If we give three "what-if" values to the above three items then, taken together, these 3*3*3*252 modes*inputs would require nearly 7000 different simulations. This is an issue since, using standard multi-objective optimizers such as NSGA-II [28], our models take seven hours to reach stable minima. Hence, using standard technology, these 7,000 runs would take 292 weeks to complete. **Note that this complaint applies not only to NSGA-II but to all the other optimizers listed above.**

In principle, such long simulations can be executed very quickly on modern CPU clusters. For example, using the NASA Ames multi-core supercomputers, the authors once accessed 30 weeks of CPU in a single week. Assuming access to the same hardware, our 7,000 runs might be completed in under ten weeks.

The problem here is that hardware may not be available. The example in the last paragraph (where 30 weeks of CPU time was accessed in one week) was only possible due to a high priority issue in need of urgent resolution. In the usual case at NASA, researchers can only 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 that assess the potential impacts). At those times, researchers can access zero CPU for any other purpose.

From the above, we say that an ideal MOEA for reasoning about complex cognitive models does so *without* incurring a high computational cost. One reason we endorse the GALE algorithm for this task is that it reaches its conclusions using far fewer CPU cycles than other approaches.

4.2 Brittleness

Ideally, any insight we gleam 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 MOEA algorithms generate "brittle" decisions.

According to Harman [36], understanding the neighborhood around individual solutions is an open and pressing issue. 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 algorithm of Figure 4. 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 [37].

Harman comments that understanding the neighborhood of our solutions is an open and pressing issue in this style of reasoning. He says:

> "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." [37].

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 perturbation stops changing objectives;
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.

5 GALE: Less Brittleness, Less CPU

This section is a brief introduction to GALE that focuses on how the algorithm tackles excessive CPU demands and brittleness. For full details, see [7].

GALE combines (a) the neighborhood perturbation (described above) with (b) the MOEA algorithm of Figure 4. The algorithm reflects over a *population* of points, each of which contains *decisions* (some inputs to the CDA model). It then searches for the input decisions that lead to best outcomes. For example:

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. [38]. Note also that GALE implements step 1 via the FASTMAP [39] 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 4, 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 [40] that builds a piecewise linear approximation to the Pareto frontier [41]. 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 [42].

Fig. 5: Inside GALE

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

Figure 5 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 having any new effect (i.e. they stop generating better objective scores). That is, all GALE solutions are guaranteed not to be brittle.

In terms of reducing runtime, the key feature of GALE is that, unlike traditional MOEA methods such as NSGA-II [28], GALE does not automatically generate objective scores for all N decisions. Instead, as it recursively clusters the data in two (using steps 1,2,3,4,5 in Figure 5), 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 $2N$ comparisons required by traditional methods like NSGA-II.

In practice, this makes GALE run much faster than traditional optimizers. Standard MOEA algorithms such as NSGA-II require 3000 to 5000 evaluations to explore the CDA model. GALE, on the other hand, performs the same task using 25 to 50 evaluations [7]. 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 repeats may be practical for GALE but highly impractical for standard optimizers. For example, twenty repeats of CDA takes 1.5 and 100 hours for GALE and NSGA-II, respectively [7].

6 A Case Study: CDA and GALE

This section describes what was learned when GALE was applied to CDA.

6.1 Background

Sometimes, when monitoring cockpit instruments, a pilot is unable to complete all 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 situation where one pilot must monitor a plane at the same time as watching over another, less experienced, pilot.

6.2 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.

6.3 Methods

This study uses the CDA model described above. 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.

For this study, GALE was run using parameters found to work best on several other models [7]:

- 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.

6.3.1 Experiments

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 [1], 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.

6.3.2 Sanity Checks

We define two sanity checks on our results:

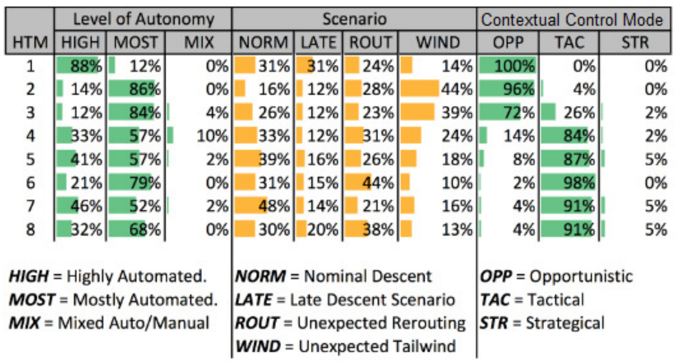
Sanity check #1: The clear pre-experimental intuition is that, as we *decrease* HTM (maximum human task load), we should see *increasing* adverse flight operations. If this observation was not seen in the results, the entire investigation should be doubted.

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.

6.4 Results

Figure 6 and Figure 7 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.



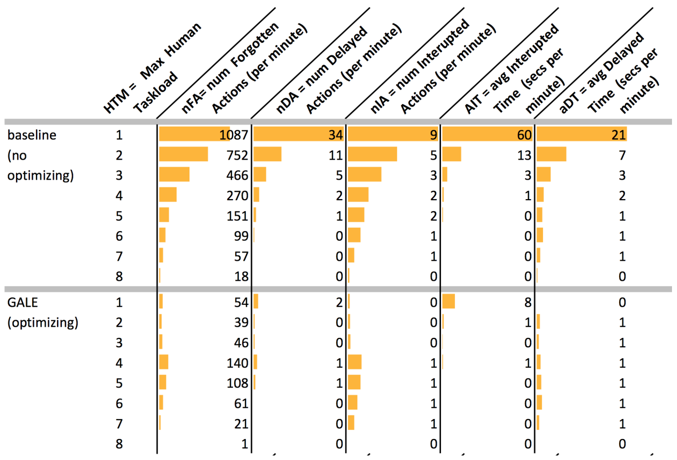
HTM	Level of Autonomy			Scenario				Contextual Control Mode		
	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	88%	12%	0%	31%	31%	24%	14%	100%	0%	0%
2	14%	86%	0%	16%	12%	28%	44%	96%	4%	0%
3	12%	84%	4%	26%	12%	23%	39%	72%	26%	2%
4	33%	57%	10%	33%	12%	31%	24%	14%	84%	2%
5	41%	57%	2%	39%	16%	26%	18%	8%	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 6a: Exp #1. Decisions found by GALE, all cognitive 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 6b: Exp #1. Objectives obtained, all contextual control modes enabled.

Fig. 6: Experiment #1: All contextual control modes enabled. Forgotten, delayed, and interrupted actions are reported by the simulation at each 0.05s time step. As an example, the same monitoring action can be 'forgotten' 20 times each second.

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*.

6.4.1 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 6b and Figure 7b.

In terms of a pilot maintaining safe operations, saturation is highly undesirable. At saturation, a pilot is permanently interrupted for all tasks so everything gets delayed. Note that this saturation result satisfies one of our *sanity check* (that less HTM leads to more problems).

The good news is that saturation can be avoided. In Figure 6b, we see that the simulated pilots following GALE's advice never reach saturation at $HTM = 1$. GALE advises the simulated pilots restrict themselves to only the most important tasks (in CDA's model, this means operating in *opportunistic* mode) and allow the automation to handle all or most of the taks 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).

6.4.2 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.

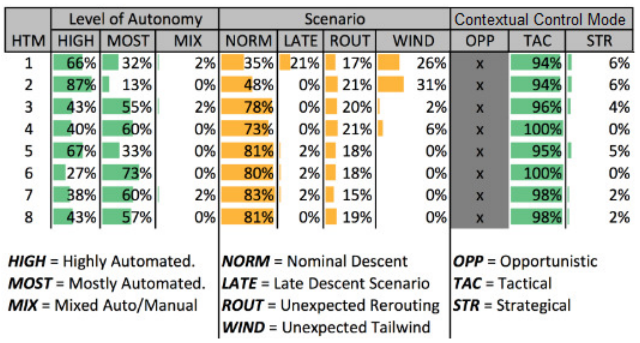
The exception to this trend are the GALE results of Figure 6b. In those rows, GALE could learn mitigations that compensate for pilots struggling to control. Those mitigations are shown in Figure 6a.

6.4.3 Absence

Several columns in Figure 6a and Figure 7a 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 7a, 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 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 7b, 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 *Scenarios* (see all the non-zero numbers in the *Scenario* columns of Figure 6b).



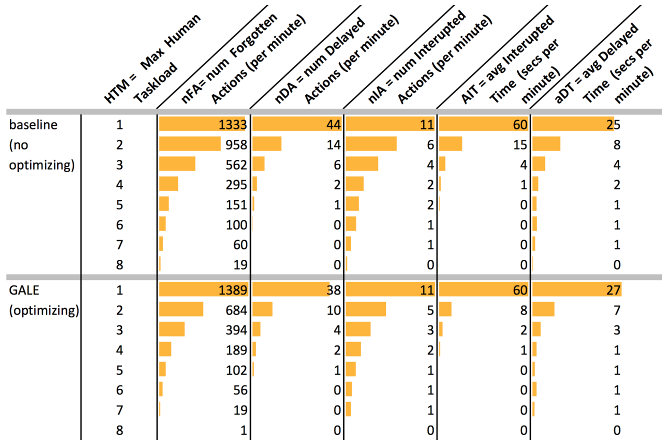
HTM	Level of Autonomy			Scenario				Contextual Control Mode		
	HIGH	MOST	MIX	NORM	LATE	ROUT	WIND	OPP	TAC	STR
1	66%	32%	2%	35%	21%	17%	26%	x	94%	6%
2	87%	13%	0%	48%	0%	21%	31%	x	94%	6%
3	43%	55%	2%	78%	0%	20%	2%	x	96%	4%
4	40%	60%	0%	73%	0%	21%	6%	x	100%	0%
5	67%	33%	0%	81%	2%	18%	0%	x	95%	5%
6	27%	73%	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 7a: 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 7b: Exp #2. Objectives obtained, opportunistic mode disabled.

Fig. 7: Experiment #2: Opportunistic mode disabled. Please see notes on Figure 6 for a description of how the tasks are counted.

6.4.4 *Cliffs*

Cliffs are values that change sharply between one HTM value and the next; there are two large cliffs in Figure 6.

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 6a. 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.

6.5 Experiment #2: Disabling Opportunistic Mode

As shown in Figure 6a, 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 7. In those results, the following aspects are noteworthy:

- Comparing the *baseline* and *GALE* distributions of Figure 7b, 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 7a to Figure 6a, 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.

6.6 On the Merits of GALE

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 discussing 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.

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.

6.7 Threats to Validity

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.

In defense of our tool selection, we note that these are the products of years of research and much analysis and testing [1]–[8], [21], [22]. 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.

As to GALE, this algorithm uses a guided $log(N)$ heuristic to explore a space of $O(2N)$ possible comparisons (for details on this approach, see §5). A pre-experimental concern with such an approach is that that this partial explanation of such a large space might miss some important details. Hence, this issue was extensively explored, using two dozen models, in Krall's thesis [7]. In short, compared to $O(2N)$ algorithms such as NSGA-II [28], GALE rarely performed worse and often performed better.

In summary, there is enough prior work on these tools to make the case that it useful to study the CDA model with the GALE optimizer.

7 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, 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.

7.1 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 to aircraft.

7.2 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 [20].

7.3 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.

8 Future Work

In the future, more studies are planned to affix the scenario of CDA and to track other objective safety metric outputs such as flight deviations to altitude, speed, and position. The version of the CDA model used in this paper currently simulates the same type of aircraft and flight route in every approach situation; there are hundreds of other options in just those two inputs to the system. When simpler aircraft and routes are being utilized, the same cognitive limitation stories may not hold.

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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

[10] 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

[11] 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/

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

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

[14] 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.

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

[16] "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.

[17] "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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] E. Hollnagel, *Human Reliability Analysis: Context and Control*. London: Academic Press, 1993, pp. 159–202.

[24] P. Zave, "Feature interactions and formal specifications in telecommunications," *Computer*, vol. 26, no. 8, pp. 20–28, 1993.

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

[26] 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.

[27] 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.

[28] 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.

[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] M. Harman and B. Jones, "Search-based software engineering," *Journal of Information and Software Technology*, vol. 43, pp. 833–839, December 2001.
[37] 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.
[38] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, January 1991.
[39] 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.
[40] S. Dasgupta, "Analysis of a greedy active learning strategy," *in Neural Information Processing Systems 17:*, vol. 1, no. x, 2005.
[41] M. Zuluaga, A. Krause, G. Sergent, and M. Püschel, "Active learning for multi-objective optimization," in *International Conference on Machine Learning (ICML)*, 2013.
[42] 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.

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

The present version of the manuscript is, I believe, close, but not quite at the point where it is ready to be published. Reviewer 2 indicates that you have addressed all of his previous issues. Reviewer 1 has a few very minor issues still to be addressed. In contrast, Reviewer 3 identified many more issues to be addressed. However, I believe that all of the issues, even the ones that Reviewer 3 can be handled with a very straightforward revision. Most of the requested revisions involve clearer definitions, updates to information, and improved wording. None require new analyses or reconceptualization of any section of the document. Having noted that these revisions should be straightforward, let me also point out that they will improve the readability of your manuscript.

Thank you for these comments. As you say, the revisions for Reviewer1 and Reviewer2 are a simple matter.

As to Reviewer3, we really should have been clearer in the last draft on many points (see our remarks, below).

Note that all significant changes to this version are shown in blue.

8.1 Reviewer1 Comments

Some more details about the case study implementation would be useful to facilitate the analysis of the results.

Particularly, the impact on the recommended actions and GALE optimization results, between the different Cognitive Control Modes should be provided.

Some details about the comparison of GALE with standard optimization methods (NSGA-II) would be important to support the claimed research contribution.

We have added text throughout the paper on the case study implementation, and we have also included additional citation information for those who would like more details.

We have paid special attention to a clearer description of GALE's recommendations, particularly as they relate to the Contextual Control Modes. (Per another reviewer's comment, we have more clearly recognized Hollnagel's contribution to the model by changing our terminology. Cognitive Control Modes have become Contextual Control Modes throughout the paper.)

Thank you for the remark on the comparison of GALE to other optimization methods– the previous version was weak on the connection of this approach to other MOEAs. We have added some notes on this topic on page 5 (please see the BLUE text there).

8.2 Reviewer 2 Comments

Since this was a collection of papers that support the use of model-based analysis of human factors for safe aircraft approach the weakness that a reader may find is that there are parts in the later two papers that are repeated and/or glazed over. This is understandable with a compilation of three papers that were all designed initially to stand independent of each other. Without combining the entire collection of papers into a single work there is little that can be done to mitigate this redundancy regardless of its effect on the reader.

Thank you for your understanding. As we said on page 1, this is not the paper that offers full details on GALE and we suspect that, like you, the interested reader will read on to the other papers.

This title paper (first in the collection) did have a few grammar mistakes that are recommended to be resolved prior to the publishing of the piece.

- Paper 1 - Section 6.3.1: [Col 1—Line 34]: where should be were
- Paper 1 - Section 6.3.2: [Col 1—Line 45]: then should be where

Typos fixed. Thank you for pointing those out.

8.3 Reviewer3 Comments

I recommend the following:

1) Clearly define and operationalize your decision variables / inputs HTM: It is not clear what HTM is presumably it stands for Human Task Load Maximum (but please confirm). It is described as an integer value ranging from 1 to 8, in which its value describes how many tasks can be handled at once before there are incurred delays, errors, or the possibility of the task being forgotten or lost. However, the number of pilots tasks can handle at once depends on the task and the resource modality. For example, I can do two things at once if they require difference resources (i.e. view a display and manually control aircraft), but I cannot do two tasks at once if they require the same resources (i.e. make two different verbal callouts at the same time). How is this accounted for in the model? I dont think any human operator can do 8 tasks at once . Maybe you should define what you mean by task and at once.

HTM was defined both in Figure 2 as an acronym for this paper, and also as one of the decision variables (in bold) in the description of CDA in Section 3. You are correct that it stands for maximum human task load. We have expanded the definitions of the four decision variables, which will hopefully lead to less confusion for future readers.

Per your recommendation, we have also made it more specific as to what we mean by 'task', and have provided the reference for the in-detail description (which takes many pages in the original reference).

Functional Allocation. The distinction among the three categories of FA (highly, mostly, and mixed) are unconventional (from the literature) and not clear. For example, in mixed-automation which tasks were handled by the computer and how is that determined? It is not clear how Highly and Mostly are different. I am confused by the wording processes the instructions vs. handle the task. Maybe some examples would help. How can the computer process flight instructions in the highly condition, if they were not first programmed by the human. And then, how is that different than mostly automated? Not clear what is the difference between Mostly and Mixed. What are examples of instructions that the computer handles in Mostly that the human handles in Mixed?

We have added text to make it clear what we mean by these function allocation levels, and also have added references for the interested reader.

The definition of the CCMs are unconventional I assume you are using Hollnagels Contextual Control Model as the basis for your work, despite that it is not referenced and you call it Cognitive Control Modes, not Contextual Control Modes. As an example, you define Tactical as Cycling through most of the available tasks and double-checking some of the computers tasks. where as, Hollnagel defines it as follows: Tactical is characteristic of situations where the persons event horizon goes beyond the dominant needs of the present, but the possible actions considered are still very much related to the immediate extrapolations from the context. (Hollnagel, 1993, pp. 170) . How do you reconcile the difference in definitions? How do you operationalize most and some, how does the model decide with tasks to cycle through and which to double check, and what is the difference between cycling and double-checking?

You are correct. The implementers of WMC used Hollnagel's work as a basis for the model. We were working from a thesis that referred to these as cognitive control modes, but they are based on Hollnagel's Contextual Control Modes. We have cleaned up the text to reflect this, and have also provided references to texts in which the exact implementation of these within WMC is described.

Section 6.3.1, page 7. Pilots made their own estimation on what where[sic] the most critical instruments to watch, and ignored all other. Since there were no actual pilots in this experiment, and it was a modeled-pilot was making the decisions what criteria did the model use to decide what instruments to monitor? For Example, Chris Wickens has a 4 parameter model called SEEV, which suggests pilots consider Salience, Expectancy, Effort, and Value what did your model use?

You are correct that we should have been more specific – the CDA model ranks the priorities of the monitoring tasks. We have summarized the model assumptions and provided a reference.

2) Clearly define your 5 Objectives Modeling whether tasks are delayed, interrupted, or dropped, is a complex problem with several different possible strategies (first in-first out, priority, criticality, salience etc). How do you operationalize forgotten actions. Does your model have a memory component? Do you model long-term, short-term memory? Do you use linear or exponential decay curves? How do you operationalize interrupted tasks. Is there a prioritization scheme. Do you use first in first out, last in first out, prioritize by importance, criticality? There is no discussion as to how this is modeled. And since this is a major part of the output, it is hard to make sense of the output.

We've expanded the objective description, briefly discussed the priority scheme and memory model, and added citations.

3) Some clarification needed in Results Figure 6,page 8. I am not understanding Number of Forgotten Actions per minute. How can an operator forget 1087 actions per minutes? How can there be 1087 actions in any given minute to be forgotten? Please define what you mean by action.

The same task can be 'forgotten' at each time step – many times in one minute. We've added a description to the captions of both Figure 6 and Figure 7.

Section 6.4.2 (bottom of section). The good news is that saturation can be avoided. In Figure 6b, if the simulated pilots following GALESs advice, they never saturate at HTM=1. It is unclear what GALEs advice was or how a pilots would follow it. In other words, you claim to have a found a mitigation that would be useful for pilots but what is it????

We claim to have found a mitigation that is useful for the simulated pilots, not real pilots. We will leave it to others (or perhaps to future work) to validate and to improve the WMC CDA model. At HTM1, in figure 6a, you see that GALE decided that the simulated pilots should fly with high levels of autonomy and in the opportunistic contextual control mode. The Scenario, for this case, mattered less (although it looks to me like the tailwind was harder to fly). We believe the interpretation of the figure 6a is obvious, and that the results are consistent with common sense.

Section 6.4.3 .. GALE could learn mitigations that compensate for pilots control struggles. Those mitigations are shown in Figure 6a

- How do we know which of those mitigations in 6a are having the effect ?
- How is Scenario a mitigation?
- How is CCM a mitigation you cant program a human to behave in opp, tac, or str modes.

1st item: Level of autonomy, scenario, and CCM are inputs to this simulation. You can see, in 6a, which decisions GALE chose. In 6b, you can see the results of those decisions on the output. 2nd item: GALE suggests: if you can only hold one item in memory, avoid flying with a strong tailwind. GALE can suggest this because the Scenario is an input to the simulation. The simulation is not reality. Also note that, even in the simulation, the Scenario mattered relatively little to the simulation. As one example, GALE chose the normal and the late scenarios equally often in its optimization for HTM=1. We do plan on future work in which we fix the Scenario (you are correct that, naively, this should more closely represent reality), and identified that with the first sentence in Section 8. 3rd item: We respectfully disagree that you can't program a human. We believe that this programming is called *training*.

4) Carefully review your conclusions and tie your

model results to practical, mitigations that can actually be implemented.

For example, on Page 10, top left From these results, we say that opportunistic mode is an essential tool for combatting the problems associated with low HTM. How is OPP a tool? You cant hand it to a pilot and tell them to use it. CCM is a result of the context / environment (e.g. if pilots are not fatigued, well-trained, carrying out a nominal descent in good weather, they can act opportunistically, yet when one more of these factors are applied, the pilot shifts to tactical or strategic modes). In other words CCM is a RESULT of the factors you model (taskload, off-nominal events), it does not MITIGATE them.

We believe very strongly that tying our results given the simulation to recommendations that should be implemented in reality is overreaching. The simulation still needs more validation. While the creators of the WMC framework are working very hard both to validate and to improve their models, as evidenced by their impressive recent results and publications, it is our understanding that there is no current computer model that is well-validated enough to be used to provide recommendations to real pilots. Such models can only suggest experiments to be performed with pilots in-the-loop.

OPP is a mode that can be can be selected as an input to the simulation. Our results showed that if the pilot had a working memory of one, they should switch to an opportunistic control mode (within the simulation). If the simulation has a direct correlation with reality, the analagous action would be to train the pilot to operate in this way: let the computer handle most of the flying (high level of autonomy), and switch to opportunistic mode (just pay attention to a few key actions). We believe such modes of operation can be trained. CCM and Autonomy level are definitely mitigations in the context of the simulation. While we have just speculated for your benefit as to how they could also be mitigations in the context of reality, we emphasize our earlier point about the necessity of further validation.

Concluding that low HTM levels are especially dangerous to aircraft is not particularly helpful. What have we learned from this that will make aviation safer? We shouldnt hire pilots that can only handle 1 item at a time? I think what we need to answer is: What do we know about the context that results in pilots not being able to handle more than 1 task at a time?

We agree that the question you posed is a very important research question to solve, and likely to be much more valuable from an aviation safety perspective than the one we address in our paper. Your question is beyond the scope of this paper, and beyond the model's ability to capture (according to our understanding).

Saying that MIX is almost never a good approach is over-reaching. MIX as modeled was not good, but it was not clear how it was modeled. Maybe the model had the computer do the wrong tasks?

Our results must be interpreted in the light of how each of these options were modeled in the simulation. Again, we are not stating that these are good results for reality (yet). In order to make recommendations about reality, we need an extra validation step.

Page 8, bottom left. Concluding Pilots should avoid cycling through all of the available monitoring tasks while trying to anticipate future tasks Calls into question the validity of this study. Do you seriously anticipate that being part of an airlines training protocol: Do not act strategically.

This study is not (yet) about reality. Note that, in future work, we plan to measure flight deviations and other safety-related output metrics. Our results say, for this model, if you want to minimize forgotten and interrupted actions (and associated delays) then don't cycle through all of the tasks and anticipate future ones. Once we have added other safety metrics, I expect the landscape of what you should do will become more complicated, even within the context of the model alone. Personally, we think the fact that we automatically discovered that high levels of autonomy and an opportunistic mode of operation decrease the number of forgotten and delayed actions appeals to common sense and makes the study more likely to valid – certainly in the context of the simulation and perhaps in the context of reality, as well.

Figure 9, bottom left. I think it is a bit of a leap to conclude: when cognitively limited pilots should trust their machines. The decision to trust a machine or not should be based on properties of the automation that make is trust-worthy (e.g. reliability, false alarm rate etc). One should have no reason to trust automation more when they are cognitively limited than when they are not cognitively limited.

Our notes above on the scope of our effort (e.g. results apply only to this simulation, and only for the output metrics we listed) apply here as well. That being said, we have softened our language throughout the paper to make it clear that we are not overreaching our scope.

5) Relate your model results and conclusions back to the two case-studies your presented in the intro. It is not clear to me how your analysis of workload, cognitive control modes, or function allocation could either explain or predict either the Air France or Asiana accidents you cite as motivation in your introduction. Can you tie these back into the discussion? You attributed the Asiana accident to pilots mis-understanding the automation. It is not clear that any of your variables taskload, autonomy levels, cognitive control modes shed any light on whether a pilot misunderstands information.

You are correct that we should have more clearly tied our results back to our motivation. We have done so. You are also correct that none of our variables can predict whether a pilot misunderstands information. We have made our scope clearer throughout the paper.

And some odds and ends: Please revisit your review of the Asiana Flight 214 crash that starts on page 2 and determine if it can be updated since more information is known now about that accident.

Thank you. We have done so.

Page 6, Section 6.1 and 6.2. You set up your study with a goal of understanding pilots lack of attention this wording puts blame on the human operator which is not merited By calling it lack-of-attention, you seem to be saying if only pilots could just pay more attention, aviation would be safer Rather, in the examples you provide in section 6.1, the pilots attention is appropriately redirected to attend to an emergency situation or unexpected flight situation .

It is hard to argue that a pilot's attention (or lack thereof) has no impact on safety. There are certainly many other attributes that affect safety, but these attributes were beyond our scope. We have tried to be more clear about our scope. We would also like to argue that, in the examples we provide, we believe the pilots attention may be *inappropriately* redirected. One possible interpretation of the Asiana 214 crash is that the pilots were so worried about unusual and yet second-order-importance tasks (e.g. training) that they failed to monitor the most important flight state variables (airspeed and altitude).

Section 4.1 You compare model-based approaches to traditional case-studies. It is not clear what you mean by traditional case studies. Do you mean pilot-in-the-loop studies? What is it about traditional case-studies that is expensive, time-consuming and difficult to reproduce?

We have changed the wording in the section to make it more clear.