Part 1: Table of Contents

Part 1: Table of Contents 4
Part 2: Identification and Significance of the Innovation 5
Part 3: Technical Objectives 12
Part 4: Work Plan 13
Part 5: Related R/R&D 15
Part 6: Key Personnel and Bibliography of Directly Related Work 18
Part 7: Relationship with Phase II or Future R/R&D 21
Part 8: Company Information and Facilities 22
Part 9: Sub-contracts and Consultants (Incl. Signed Commitment Letters) 23
Part 10: Commercial Potential Applications 24
Part 11: Similar Proposals and Awards 25

Part 2: Identification and Significance of the Innovation

Summary: Quality vehicle health management systems are critical to the successful operation of modern sounding rockets, and other unmanned vehicles. Unfortunately, the software of these systems tends to be complex and rigid and thus expensive and failure-prone, especially given the several real-time constraints of rocketry. We propose to develop the "Magic Bullet" Adaptive Intelligent Vehicle Health Management (AIVHM) System, a novel adaptive control system for sounding rockets based on the technologies of treatment learning and Bayes classification. This system will be able to derive an appropriate control strategy for a vehicle in the event of partial system failure. Our relationship with the Portland State Aerospace Society (PSAS) provides us with a unique opportunity to evaluate and deploy these methods at extremely low cost and with extremely low risk, for simulation and even actual flight testing. The PSAS LV2 rocket has a navigation and control system architecture ideally suited to experimentation with the proposed system. As senior technical advisory to PSAS, our organization is well-positioned to prototype and deploy the Magic Bullet AIVHMs technology with PSAS. We expect this deployment to result in the information needed to scale the technology to larger, more complex, more demanding avionics applications.

As widely-available hardware grows more capable and sophisticated, the systems engineering problem changes: the hard part today is to build software controllers for these systems. Even a small rocket, for example, tends to have a large amount of complex and error-prone code for navigation and guidance. Methods from machine learning and data mining can significantly reduce this systems engineering cost.

We propose to construct an AIVHM system suitable for use in unmanned vehicles. An AIVHMS can improve the odds of safely completing missions by managing control strategies during operation. We believe that the costs of building an AIVHMS can be significantly reduced by using automatic data miners to learn vehicle health policies. We expect our proposed AIVHMS to be ultra-fast, accurate, and adaptive enough to solve complex VHM problems in real-time.

Figure 1: This Trident missile shows the consequences of complex, inadaptive avionics software.

Most data miners run too slowly to be successful in real-time AIVHM applications. Our Magic Bullet AIVHM represents a new kind of *real-time anytime data miner*. Unlike standard learners, Magic Bullet rapidly, incrementally, and continuously improves its theories as new data arrives. That is, Magic Bullet can offer recommendations on how to improve vehicle health at any time, and can find better recommendations as time progresses. Magic Bullet is a *treatment learner* [20] that reports the *least* that can be done to *most* improve the current vehicle situation. Treatment learners are especially useful in time-critical situations, when operators only have time to perform a limited number of actions.

The current generation of treatment learners are too slow for real-time AIVHM. However, using Bayesian methods, it may be possible to speed up treatment learning 100-fold. Treatment learning will then become the basis of a novel, easy-to-build IVHM technology. The alternative to Magic Bullet is a purpose-built (hence expensive) IVHM that deals poorly with entirely unforeseen circumstances. Magic Bullet, on the other hand, will be able to handle the unforeseen by learning new solutions on-the-spot.

Our plan is to experiment with the Magic Bullet AIVHMS on an existing ultra-low-cost sounding rocket (see §2.1). In Phase I of the proposed work, we will construct software simulations of a target sounding rocket, implement the Magic Bullet AIVHM learner for the rocket (see §2.2), and evaluate Magic Bullet in simulation. In Phase II, we will test our software on the actual target vehicle. Error conditions deemed recoverable in simulation will be deliberately induced in the rocket during flight. It is anticipated that some launch vehicles will be damaged beyond repair during this process. However, the extremely low parts cost of the target vehicle (less than $5,000) makes this sort of testing a reasonable proposition.

2.1 The LV2 Rocket

Our test rocket is the Portland State Aerospace Society (PSAS) LV2 rocket. Amateur rocket groups such as PSAS have a rich history of significant contributions to science and engineering. Indeed, most national space programs can trace their roots back to small, active groups of amateurs working on rocketry during the early part of the 20th century [30]. Over its 5-year history, the Portland State Aerospace Society has established a reputation as one of the world's foremost amateur rocketry groups. The group is the first to have the honor of being featured as "IEEE Innovators", in a recent IEEE advertising campaign. The open-hardware, open-software, open-development model used by the team has led to widespread replication of its work. PSAS was the featured exhibitor at the Usenix 2003 Annual Technical Conference.

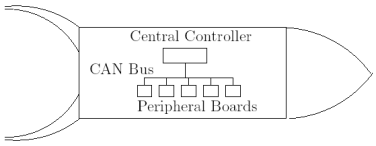


Figure 2: PSAS LV2 Avionics Architecture

LV2 is the 4^{th} generation PSAS sounding rocket and is a non-trivial combination of avionics hardware and software, including a six-axis inertial measurement unit, on-board GPS, digital telemetry down link, and amateur TV display as well as more traditional avionics systems. Primary flight sequencing is autonomous, with several levels of backup. The rocket flies at speeds up to Mach 2.4, with a design altitude of 45,000 feet (14 miles): it is controlled by an on-board Debian GNU/Linux flight computer [24].

LV2's active guidance system is rapidly evolving. The current design has lateral thrusters for roll control and oxidizer injection thrust vector controllers for pitch and yaw control. The design includes a bell-shaped chamber below the nozzle of LV2 containing four oxidizer injectors. Injecting oxidizers into that chamber creates a secondary burn that produces off-axis unbalanced thrust.

The avionics architecture of the PSAS system is shown in Figure 2. A central PC-104 flight computer with an AMD Elan x86 processor communicates with a number of peripheral boards via the high-reliability Controller Area Network (CAN) serial bus. The peripheral boards contain a Microchip PIC micro-controller together with dedicated hardware functionality such as inertial measurement, ignition, and video telemetry. A large amount of rocket and ground software (valued at about $1M in commercial development—see §6.3) controls the avionics hardware.

A key feature of Figure 2 is the separation of the controller (a Debian/GNU LINUX system) from the rest of the rocket. The narrow interface achieved by this separation makes it straightforward to construct a reasonably detailed launch vehicle simulator for this craft, permitting more extensive and extreme experimentation with new ideas such as Magic Bullet without risking even the extremely inexpensive hardware used by the PSAS team.

During the last flight of LV2, the fin canister separated from the airframe around Mach 1 due to a defect introduced in a last-minute configuration change. Nonetheless, the rocket flew onwards for several seconds before becoming unstable and crashing. For a nearly stable vehicle such as LV2, it is quite possible that active guidance by an on-board AIVHMS could have taken the rocket to a recoverable flight configuration. Even better than saving the vehicle, it is possible that the mission could have been completed in this circumstance.

There are several ways an AIVHMS can use active guidance to compensate for in-flight emergencies in this system. For example, a persistent steering error due to airframe failure might be resolved by placing the rocket in a spin using the lateral thrusters, or by simply using the thrust vector controllers to counteract the error.

Our intent is to specify a set of *skeletal repair plans* (e.g. spin rocket, inject oxidizers) with slots for parameter settings (e.g. spin rate, which oxiders to use, rate of oxidizer injection). An AIVHMS would then have two tasks: selecting which skeleton to apply and learning appropriate slot parameters.

For such on-board AIVHMS to be practical, the IVHM needs an ultra-fast real-time learning algorithm for skeleton selection and parameter tuning. Our proposal is to use *treatment learning*.

2.2 Treatment Learning

Treatment learning is a data mining method invented by one of us (Menzies) using funds from NASA's Office of Safety and Mission Assurance[1]. Treatment learners find contrast sets that separate different outcomes [18, 20]. Unlike standard contrast set learners (e.g. STUCCO [1]), treatment learners seek *minimal treatments*; i.e. the *smallest* sets that *most* separate outcomes. Treatment learners hence have a potential advantage over standard learners for learning control strategies, since the treatment learner will report the fewest changes needed to improve the current situation.

Treatment learners build up their treatments by combining *promising* attribute ranges. A range is promising if it *lifts* the distribution of outcomes; i.e. it *selects* outcomes we like and *avoids* undesirable outcomes. For example, consider the log of golf playing behavior shown in Figure 3 that shows golf play over 14 weekends under various weather conditions. The *baseline* golf playing behavior comprises playing lots of golf six times, some golf three times, and no golf eight times; i.e.

outlook	*temp(°F)*	*humidity*	*windy?*	*class*	
sunny	*85*	*86*	*false*	*none*	✘
sunny	*80*	*90*	*true*	*none*	✘
sunny	*72*	*95*	*false*	*none*	✘
rain	*65*	*70*	*true*	*none*	
rain	*71*	*96*	*true*	*none*	
rain	*70*	*96*	*false*	*some*	
rain	*68*	*80*	*false*	*some*	
rain	*75*	*80*	*false*	*some*	
sunny	*69*	*70*	*false*	*lots*	✘
sunny	*75*	*70*	*true*	*lots*	✘
overcast	*83*	*88*	*false*	*lots*	✔
overcast	*64*	*65*	*true*	*lots*	✔
overcast	*72*	*90*	*true*	*lots*	✔
overcast	*81*	*75*	*false*	*lot*	✔

Figure 3: A log of golf-playing behavior.

$$\frac{5}{14} \cdot \textit{none} + \frac{3}{14} \cdot \textit{some} + \frac{6}{14} \cdot \textit{lots}$$

where $\frac{5}{14}, \frac{3}{14}, etc$ is the *probability* of each class in the baseline (i.e. original) data set (denoted $P(class \mid baseline)$). Suppose we know the *utility* of each class (i.e. how much we value each outcome) is *none* $= 2$, *some* $= 4$, *lots* $= 8$. We can then sum this baseline as

$$\textit{sum}(\textit{baseline}) = \frac{\sum (P(\textit{class} \mid \textit{baseline}) \cdot \textit{Utility}(\textit{class}))}{\sum \textit{Utility}(\textit{class})} = \frac{\frac{5}{14} \cdot 2 + \frac{3}{14} \cdot 4 + \frac{6}{14} \cdot 8}{2+4+8} = 0.357 \quad (1)$$

These scores model the domain-specific view of the relative merits of playing lots of golf. (The exact scores do not matter too much, just as long as we normalize their sum as we have above.)

Consider the effects of applying the decision not to play golf on rainy or sunny days. In effect, this means "treating" the data by selecting the ticked examples in Figure 3 where $outlook = overcast$. The "treatment" $outlook = overcast$ yields four examples with lots of golf played; i.e.

$$sum(\textit{outlook} = \textit{overcast}) = \frac{\sum (P(\textit{class} \mid \textit{outlook} = \textit{overcast}) \cdot \textit{Utility}(\textit{class}))}{\sum \textit{Utility}(\textit{class})} = \frac{\frac{0}{4} \cdot 2 + \frac{0}{4} \cdot 4 + \frac{4}{4} \cdot 8}{2+4+8} = 0.57 \quad (2)$$

The lift can now be calculated: it is the ratio of the sum of the treatment to the sum of the baseline:

$$\textit{lift}(\textit{outlook} = \textit{overcast}) \quad = \quad \frac{\textit{sum}(\textit{outlook} = \textit{overcast})}{\textit{sum}(\textit{baseline})} = \frac{0.57}{0.357} = 1.6 \quad (3)$$

In the language of treatment learning, promising attribute ranges are those that result in lifts greater than one; i.e. most *improve* the distributions of the outcomes when compared to the baseline. Treatments are formed by combining these promising attribute ranges. The golfing data set is tiny: there is little to be gained there by considering complex combinations. In fact, the best treatment for the golf example is a single attribute range: $R_X(\textit{outlook} = \textit{overcast})$.

[1] 2000–2003, West Virginia University Initiatives

Looking at Figure 3, is simple to see why this treatment is so effective: $R_X(\textit{outlook} = \textit{overcast})$ appears *always* when lots of golf is played and *never* otherwise.

Consider the golfing decision as a control problem. Treatment learning finds an accurate, simple decision method (in this case, noting whether *outlook* = *overcast*) for a control output (in this case, whether to play golf).

Treatment learning can also learn *minimal monitors* that can alert the operator to hazardous conditions. For example, by reversing the scores on our classes (to *none* $= 8$, *some* $= 4$, *lots* $= 2$) and repeating the above analysis, we find that when *outlook* = *sunny*, we decide to play less often than not. (This treatment selects the crossed examples in Figure 3: 3 *don't plays* and 2 *plays*).

Armed with the above results, we can now control our golfing. If golfing becomes a mission-critical operation, then we can raise an alert when *outlook* = *sunny*. Faced with such a golf playing crisis, we can (say) select a vacation location where *outlook* = *overcast*.

2.3 Adaptive IVHM and the Small Treatment Effect

Significantly, our treatments for the golfing problem in §2.2 were compact and simple, requiring only a few of the available attributes. In particular, we only needed *outlook*: the other attributes (*temperature, humidity, windy*) of Figure 3 were essentially useless in helping to answer our question. Analogous results to this *small-treatment effect* have been reported in other fields.[2] In many cases, what happens in an entire system can be controlled by setting values in a small critical region. Mathematically, this effect can be explained by considering the expected number of decision points in a space of connected concepts [22].

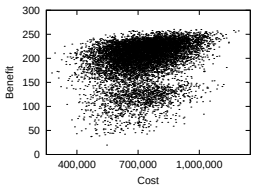
The small-treatment effect is of vital importance to practical AIVHM. Standard data miners generate intricate descriptions of the attribute ranges associated with different outcomes (e.g. J48 decision tree learner [31]). The more complex the treatment, the more effort associated with executing a repair. Conversely, the simpler the treatment, the faster it can be applied. Further, and more importantly for real-time controllers, smaller target concepts are often easier and faster to find. For this reason, treatment learners are our preferred choice for AIVHM.

2.4 Applications of Treatment Learning to NASA

Treatment learning has already been applied, to non-real-time tasks in NASA applications. Feather and Menzies [10] report experiments in the use of treatment learning to optimize requirements models for deep space missions. Cornford and Feather's DDP [2] tool allows users to sketch out mappings between requirements, risks, and mitigations. This enables search for the *cheapest* mitigations: those that *most* reduce risks while achieving the *most* requirements. For humans, finding these mitigations can be overwhelmingly complicated. For example, one deep space mission analyzed using DDP has 99 possible mitigation options, yielding $2^{99} \approx 10^{30}$ mitigation strategies to consider. This space is too large to explore thoroughly. For example, Figure 4(a) shows the results of 50,000 runs with DDP. In each run, a random set of mitigations were selected. Note the huge range of possible costs and benefits.

The large space of mitigation strategies within DDP was constrained and improved using a treatment learner that divided Figure 4(a) into *preferred* (in the sense of lower costs and higher benefits) and *undesired* regions. With knowledge of that division, a treatment learner found a set of constraints that selected for the preferred outcomes while avoiding the undesired regions. The resulting preferred treatments are shown in Figure 4(b). This space provides a range of high-quality strategies from which a controller could select either randomly or according to optimization criteria. Note that the variance in behavior was indeed greatly *reduced* while *decreasing* mean costs and *increasing* mean benefits. Significantly, and as predicted by the small-treatment effect, treatment learning achieved this result using only a small subset of the available risk mitigation options.

[2] *Master-variables* in scheduling [4]; *prime-implicants* in model-based diagnosis [27], machine learning [26], and fault-tree analysis [17]; *minimal environments* in the ATMS [5]; *base controversial assumptions* of HT4 [19]; the small *feature subset selection* effect [16] and the related *1R* effect [13].

(a) BEFORE: 

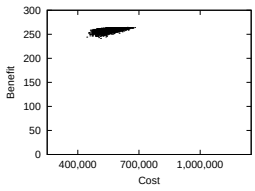
(b) AFTER: 

Figure 4: An application of treatment learning. Here, one dot is one project plan; i.e. one possible setting to the 99 risk mitigation options for a JPL deep space mission. X-axis= "cost" = sum of the cost of selected risk mitigations (lower is better). Y-axis= "benefit"= requirements coverage, less the effects of risk (more is better)

2.5 Drawbacks of Treatment Learning

One disadvantage with the current generation of treatment learners is that they cannot update their own theories when new data arrives. *Anytime* learners can offer, at any time, some current best theory: this best theory improves monotonically over time.

Another problem with current treatment learners is that they are too slow for real-time AIVHM. Our first practical treatment learner, TAR2, found treatments by computing the lift of all combinations of $1..Max$ promising attribute ranges (where *Max* was some user-defined parameters). Such a search is theoretically slow ($O(2^{Max})$). However, in confirmation of the small-treatment effect, TAR2 found effective treatments using small *Max* (*Max* ≤ 8) on its test domains [21].

A later version, TAR3 [14], built and tested treatments from N attribute ranges, where N was chosen randomly in the range $1 \leq N \leq Max$. Each treatment of size N was then filled in using attribute ranges selected in a random fashion weighted using the *lift* distribution of the individual ranges. (TAR3 selected high *lift* ranges more often than low *lift* ranges.) TAR3 ran much faster that TAR2 both in theory ($O(Max)$) and in practice, and found nearly the same treatments as did the more exhaustive search of TAR2.

Even though TAR3 is faster than TAR2, it still takes seconds to runs. In order to make treatment learning fast enough for real-time AIVHM, we need a treatment learner that is two orders of magnitude faster than TAR3. Further, it should be an anytime algorithm. We call this ultra-fast anytime treatment learner Magic Bullet.

2.6 Towards Better Treatment Learners

To build Magic Bullet, we will adapt ideas from other kinds of data miners. Of the many available data mining techniques, our research led us to three approaches that we believed best suited to our domain: *neural networks*, *instance-based learners*, and *Bayes classifiers*.

The back-propagation of neural networks [25] incrementally adjusts the weights of the network in response to how well the network predicts for system output. Hence, neural nets are *anytime* algorithms. However, while a trained neural net runs fast enough for real-time responses, training time can be too slow to allow for real-time adaption.

Instance-Based Learners (IBL) [3] have performance properties that directly contrast with those of neural nets. Training is extremely fast in IBL: new examples are just inserted into the n-dimensional space of all previous examples. However, a trained IBL system can be extremely slow in classifying a new example, as it typically must examine all previous examples in order to make a classification. A k-th nearest neighbor system, for instance, must compute the distance from the target example to every previous example in order to make a classification. While methods for improving performance are known, they tend to be complex and do not entirely alleviate the classification speed problem.

A promising approach to real-time anytime treatment learning is to handle treatment learning as an extension to

attributes			class
MAKE	SIZE	HIFI	TYPE
BMW	medium	yes	coupe
BMW	small	yes	coupe
Benz	small	yes	coupe
Ford	small	yes	coupe
Honda	small	no	coupe
Mitsubishi	small	yes	coupe
Toyota	large	no	coupe
Toyota	small	yes	coupe
Benz	large	no	SUV
Ford	large	no	SUV
Mitsubishi	medium	no	SUV
Toyota	large	no	SUV

$F =$

attributes						class
MAKE	coupe : SUV	SIZE	coupe : SUV	HIFI	coupe : SUV	coupe : SUV
Mitsubishi	1 : 1	Small	$6 : 0^a$	yes	$6 : 0^b$	8 : 4
Toyota	2 : 1	Medium	1 : 1	no	2 : 4	
Benz	1 : 1	Large	1 : 3			
BMW	$2 : 0^c$					
Ford	1 : 1					
Honda	$1 : 0^d$					

$P(E_i \mid C_j) =$

attributes						class	
MAKE	coupe : SUV	SIZE	coupe : SUV	HIFI	coupe : SUV	coupe	SUV
Mitsubishi	$\frac{2}{9} : \frac{2}{5}$	Small	$\frac{7}{9} : \frac{1}{5}$	yes	$\frac{7}{9} : \frac{1}{5}$	$\frac{9}{13}$	$\frac{5}{13}$
Toyota	$\frac{2}{9} : \frac{2}{5}$	Medium	$\frac{1}{9} : \frac{2}{5}$	no	$\frac{1}{9} : \frac{3}{5}$		
Benz	$\frac{2}{9} : \frac{1}{5}$	Large	$\frac{2}{9} : \frac{2}{5}$				
BMW	$\frac{3}{9} : \frac{1}{5}$						
Ford	$\frac{2}{9} : \frac{2}{5}$						
Honda	$\frac{2}{9} : \frac{1}{5}$						

Figure 5: A log of car types (left) and extracted statistics (right). Zero frequency entry (the cells ^{abcd} in F) would make $\prod_i P(E_i \mid H)$, and hence Equation 4, evaluate to zero. Standard practice when computing $P(E_i \mid H)$ is thus to add one to all frequency counts in F.

Bayes classifiers [7, 8]. A *Bayes classifier* tunes past knowledge to new evidence using Bayes' Theorem:

$$P(H \mid E) = \frac{P(H)}{P(E)} \prod_i P(E_i \mid H) \tag{4}$$

That is, given fragments of evidence E_i and a prior probability for a class $P(H)$, a posterior probability $P(H \mid E)$ is calculated for the hypothesis given the evidence. For example, suppose our task is to decide if a *medium*-sized *Ford* is a *coupe* or an *SUV*. To solve this problem, a Bayes classifier imports examples of *coupe*s and *SUV*s (Figure 5, left hand side) to compute the frequencies F of the attribute ranges in the different classes. Figure 5, right hand side, shows an estimate for $P(E_i \mid H)$ computed by dividing the frequency of an attribute range in a class by the frequency of that class.

The likelihood L of evidence belonging to some class is the product of the probability of that class times the conditional probability of that evidence. Figure 5 shows that $P(\textit{SUV}) = \frac{5}{13}$; $P(\textit{MAKE} = \textit{Ford} \mid \textit{SUV}) = \frac{2}{5}$; and $P(\textit{SIZE} = \textit{medium} \mid \textit{SUV}) = \frac{2}{5}$. Hence $L(\textit{SUV} \mid E)$ can be calculated and, in a similar manner, $L(\textit{coupe} \mid E)$

$$L(\textit{SUV} \mid E) = \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{5}{13} = 0.0615385 \qquad L(\textit{coupe} \mid E) = \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{9}{13} = 0.011396 \tag{5}$$

These are then normalized to probabilities as follows:

$$P(\textit{SUV}) = \frac{0.0615385}{0.0615385 + 0.011396} = 84\% \qquad P(\textit{coupe}) = \frac{0.011396}{0.0615385 + 0.011396} = 16\%$$

That is, it is more likely that *MAKE* = *Ford*, *SIZE* = *medium* is an *SUV* than that it is a *coupe*.

Many studies (e.g. [11, 6]) have reported that, in many domains, this simple Bayes classification scheme exhibits excellent performance compared to other learners. Bayes classifiers can be extended to numeric attributes using kernel estimation methods. The default numeric kernel estimation is a simple gaussian [31]. Other, more sophisticated methods are well-established [15, 9], but several studies report that even simple methods suffice for adapting Bayes classifiers to numeric variables [6, 32].

The repeated success of such simple Bayesian methods surprises many researchers. Such classifiers are often called *naïve* [31], since they assume that the frequencies of different attributes are independent. In practice [33], the absolute values of the classification probabilities computed by Bayes classifiers are often inaccurate. However, the relative ranking of classification probabilities is adequate for the purposes of classification. For example, in the above

example with medium-size Fords, a Bayes classifier might correctly conclude that $P(\textit{SUV})$ is greater than $P(\textit{coupe})$ even if the precise numeric values for $P(\textit{SUV})$ and $P(\textit{coupe})$ are inaccurate.

We conjecture that a Bayes classifier can be extended to become a fast Bayesian treatment learner. Currently, each time TAR2 or TAR3 builds a treatment T it assess that treatment via a complete pass through the database of examples to recalculate Equation 2. Note that the purpose of this pass is to count how many examples of which particular class are selected by a treatment. An alternative, possibly much faster, method is to *estimate* those counts using Equation 5. Recall that Equation 5 returns the likelihood that a given example falls into a particular class. The likelihood $L(\textit{class} \mid T)$ estimates the fraction of examples of class C selected from a database by treatment T. Multiplying $L(\textit{class} \mid T)$ by the number of examples in the database therefore gives an estimator for the number of examples of class C selected by the treatment T: this estimate is exactly the value needed to compute treatment lift. Such a Bayesian treatment learner should have several advantages, as listed below.

Achieves Real-Time Response: If $L(\textit{class} \mid T)$ is indeed an adequate estimator for the counts used by Equation 2, then candidate treatments can be assessed *without* a time-consuming pass through the database. Training such a Bayes treatment learner is fast, since it only requires updating the frequency tables. Further, classification with such a treatment learner is also fast, since there are only a few calculations required for each new example. A Bayes treatment learner can thus achieve real-time response rates.

Handles Incomplete Information: An on-board AIVHMS should be able to make decisions even if some of the sensors are inoperative. We note above in the car example that Bayesian methods can easily handle such missing information. The above example (about medium-sized Fords) made no reference to one of the attributes in the example set (presence or absence of *HIFI*). If the available evidence makes no comment on a certain attribute X (e.g., $X = \textit{HIFI}$), Bayes methods simply omit $P(H \mid E_x)$ from the product term and adjust $P(E)$ appropriately. That is, the methods use only available evidence to tune the priors. (By contrast, many other classification methods have a harder time adapting to missing data.)

Needs Little Memory: The inputs to a Bayes method are summarized in small frequency tables. This means that multiple tables can be held in main memory at once. Because these frequency tables are small, it is possible to maintain different frequency tables for different modes of operation. These modes may be pre-specified by the mission profile (e.g. launch, booster separation, etc) or automatically learned (e.g. the above example in which the fin canister is lost during the boost phase).

Achieves Anytime Inference: At any time, a Bayesian frequency table contains information on the examples seen to date. As time progresses, more examples are seen: the frequency counts are more precise and the classification accuracy improves. Better yet, if the rocket's mode somehow changes and old examples become irrelevant, a Bayes frequency table can automatically detect when it should *switch* to a new mode and collect new frequency counts. The trigger for such a switch is when previously stable distributions in its frequency tables start changing significantly. Such a change might occur when, say, the rocket's aerodynamics have changed due to airframe failure. Note that just after such a mode change, the performance of the system will drop while new knowledge is gained for the new mode. The learner is still anytime, however, since the best treatment for the new mode will always be available. As more experience is gained with that mode, its treatment will get better.

Part 3: Technical Objectives

Based on the above, several technical questions present themselves:

How Fast Is Fast Enough? The core technical question of this project is how to build an AIVHMS that is fast enough for on-board real-time control. To answer this question, some domain modeling is required to define numerous fault scenarios, their possible repairs, and the time required for those repairs.

Is Treatment Learning Fast Enough? Once we have established the performance requirements, we then need to verify that small treatments can be found fast enough to support on-board in-flight adaptation. We also need to insure that these treatments can be evaluated sufficiently quickly to guide real-time control.

Is Treatment Learning Adequate? We have some confidence that treatment learning will be useful for adaptive IVHM. However, the effectiveness of our current treatment learners depends on the small-treatment effect (see §2.3). We need to verify that complex devices like LV2 can be effectively controlled by setting a small number of parameters.

Can We Use Likelihood Estimators For Lift? Based on our analysis of Bayesian methods, we believe that using likelihood estimators for lift will speed up the treatment learner without significantly impacting its effectiveness.

Can Likelihood Rank Treatments Accurately? A Bayesian treatment learner may suffer from the *naïve* Bayes classifier assumption, namely of independence between attributes. Our pre-experimental intuition is that the fortuitous results for naïve Bayes classifiers will also hold for Bayesian treatment learners: even if the *absolute value* of the lift predicted by Bayesian methods is wrong, the *relative rankings* of the treatments will still be correct. That is, our expectation is that an ultra-fast Bayesian treatment learner will be able to infer what is the best current treatment, even if we are unable to predict the exact effects of that treatment.

Will Frequency Table Changes Indicate Mode Changes? Our Bayesian treatment learner will also need to recognize mode changes in the rocket (e.g. a fin falling off) so it can start a new frequency table for the new mode. Later on, the rocket might move back to an old mode of operation. If so, then it would be useful for the Bayesian treatment learner to retrieve cached frequency tables for the old mode for reuse. To enable this, we need to verify that Magic Bullet can effectively switch from current modes to new modes as well switching from the current mode back to old modes.

Part 4: Work Plan

All the work for Phase I will occur in Portland, Oregon. For Phase I, the work breaks up into six tasks. Initially, we will build the infrastructure needed to construct Magic Bullet, in three steps. We will (A) perform some domain modeling to (B) design and build a flight simulator. This simulator, combined with a (C) control language for the LV2 hardware, will be used to experiment with the Magic Bullet AIVHMS.

We will construct Magic Bullet in two steps. We will (D) experiment with the Bayesian treatment learning using standard public domain data sets, then (E) conduct experiments with that treatment learner using the simulator. Of course, ongoing during the course of the project will be the need to (F) manage the project.

Note that deliverables of part (E), the Magic Bullet and associated report, will be the end deliverables for this project.

The relationship between the proposed technical tasks and the technical questions are shown in Figure 6. Each task, its deliverables, and its due date is described in Figure 7. No tasks involving direct integration of the Magic Bullet AIVHMS and the LV2 avionics hardware and software are part of the Phase I plan. These activities will be deferred to Phase II, as discussed in Part 6.3 of this proposal.

	Technical Tasks (defined in Figure 7)				
Technical Questions (defined in Part 2.6)	A) Dom. Model	B) Simulator	C) Language	D) Treatment Learning	E) Simulations
1. How fast is fast enough?	✔	✔			✔
2. Treatment learning is fast enough?			✔	✔	✔
3. Treatment learning is adequate?				✔	✔
4. Likelihood estimates for lift?				✔	✔
5. Likelihood can rank treatments?			✔	✔	✔
6. Frequency tables can detect mode changes?				✔	✔

Figure 6: Relationship between tasks and technical questions.

ID	Task	Who	Hours	Notes	Deliverable	Due
Massey tasks:						
A	Perform domain modeling	Massey	100	Define the major modes of the rocket (e.g. launch, separation, deploy parachute, etc). For each mode, define nominal and off-nominal operational profiles (probability distributions for input values). For a sample of the off-nominal scenarios, construct a library of *skeleton repair plans* (actions to be performed to repair some emergency situation).	For the list of known off-nominal scenarios, definition of the required response time of an AIVHMS.	March 2005
B	Build flight simulator	Massey	200	Based on the architecture of Figure 2, replace the peripheral boards with simulators of flight data. This simulator will need the nominal and off-nominal operational profiles defined above.	Logs of simulated sensory data from the rocket in simulated flight can be collected for two selected operational profiles (one nominal, one off-nominal).	April
C	Define a control language for the LV2 hardware	Massey	150	The AIVHMS will need a high-language for querying the state of the rocket and issue control commands.	Traces of control language scripts executing, reporting sense data and setting controllable.	May
Menzies tasks:						
D	Experiment with bayesian treatment learning:	Menzies	180	While the simulator is being built, technical questions can still be explored by building a Bayesian treatment learner and testing it on public-domain data sets.	A report on Bayesian treatment learning on public-domain data sets discussing technical questions 4–6.	May
E	Conduct experiments using simulator	Menzies	230	Once the simulator and the control language are available, Bayesian treatment learning can be tested for adaptive IVHM on the LV2 simulator.	A report on Bayesian treatment learning for adaptive IVHM for the LV2 rocket, discussing technical questions 2–6.	June
F	Proposal management, contact with NASA	Menzies	40	Ensuring the deliverables arrive on time; reporting to NASA, as required.	—	—
		Total hours	900			

Figure 7: Tasks, teams, deliverables, due dates (2005 end-of-month).

Part 5: Related R/R&D

5.1 IVHM at NASA

IVHM has been previously discussed by many groups including the *Model-based Diagnosis and Recovery Group* at Ames Research Center (ARC)[3] That group bases its work on model checking of autonomous systems, based on the LIVINGSTONE model-based systems kernel [23]. In October 1998, that group launched Deep Space 1 (see Figure 8). For part of its mission, DS1 was flown autonomously by Remote Agent, an AI system combining high-level planning and scheduling with intelligent execution. If something went wrong during flight, LIVINGSTONE recognized there was a problem and could ask two levels of on-board experts for advise. An "executive" could be consulted for simple procedures that might quickly remedy the problem. If those simple procedures failed to resolve the problem, the executive gave up and passed the problem to an AI planner that generated a new plan for achieving mission goals.

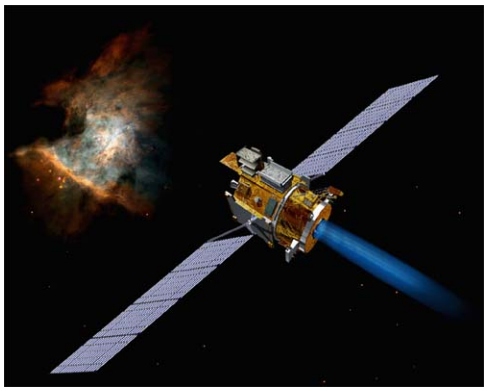


Figure 8: NASA's Deep Space One remote agent experiment.

The skeletal repair plans used in Magic Bullet are far less detailed that the models used in LIVINGSTONE. In essence, LIVINGSTONE reasons over complete models of a spacecraft while Magic Bullet reasons over partial models of repair actions. For systems that are fully described in a declarative model, LIVINGSTONE-based IVHM may be the preferred approach. However, when such complete models are non-trivial to build, the skeletal repair plans of Magic Bullet may be a more cost-effective solution.

Another advantage of Magic Bullet is the rapid response and adaptivity of the system. The general-purpose planner of LIVINGSTONE may not be quick enough to monitor and manage a sounding rocket during boost phase. Our proposed work will check our conjecture that Bayesian treatment learning is fast. If this research is successful, then a clear future direction (beyond the scope of this proposal) is to explore combining the detailed reasoning of LIVINGSTONE with the adaptivity of Magic Bullet.

5.2 Control Theory

Standard avionics control models for robust systems are often based on *piecewise-linear* control theory (e.g. [12]). The fundamental idea is that a transition from one linear control domain to another is made when the control boundaries of the operating controller are exceeded. In principle, this can be an effective method of recovering from control failure: however, the devil is in the details. The set of control domains and their associated controllers must be determined statically up-front. This is difficult: it requires envisioning the set of all possible flight regimes, including pathological ones, and deriving appropriate controllers. Magic Bullet may be able to better that, adaptively constructing control rules even for unforeseen situations.

Another problem with piecewise-linear controllers is that a decision rule must be given for each domain. The decision rule specifies the conditions for a transition to be made, and the new domain to be transitioned to. Again, this can be difficult, requiring significant nonlinear reasoning to be done up front, and expensive calculation at runtime. By contrast, Magic Bullet is specifically designed to adapt to new flight regimes on its own, recognizing automatically when the current control rules are inadequate. This strength of Magic Bullet in AIVHM should provide a significant advance in the state of the art for such controllers.

[3] See `http://ic-www.arc.nasa.gov/projects/mba/`.

5.3 Data Mining

The relationship of this work to other data mining approaches was discussed above using the following references.

References

[1] S. Bay and M. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999. Available from `http://www.ics.uci.edu/~pazzani/Publications/stucco.pdf`.

[2] S. Cornford, M. Feather, and K. Hicks. DDP a tool for life-cycle risk management. In *IEEE Aerospace Conference, Big Sky, Montana*, pages 441–451, March 2001.

[3] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, pages 21–27, January 1967.

[4] J. Crawford and A. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *AAAI '94*, 1994.

[5] J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.

[6] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[7] R. Duda, P. Hart, and N. Nilsson. Subjective bayesian methods for rule-based inference systems. In *Technical Report 124, Artificial Intelligence Center, SRI International*, 1976.

[8] R. Duda, P. Hart, and R. Reboh. Letter to the editor. *Artificial Intelligence*, 26:359–360, 1985.

[9] U. M. Fayyad and I. H. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

[10] M. Feather and T. Menzies. Converging on the optimal attainment of requirements. In *IEEE Joint Conference On Requirements Engineering ICRE'02 and RE'02, 9-13th September, University of Essen, Germany*, 2002. Available from `http://menzies.us/pdf/02re02.pdf`.

[11] M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions On Knowledge And Data Engineering (to appear)*, 2003.

[12] D. Henrion, G. Garcia, and S. Tarbouriech. Piecewise-linear robust control of systems with input saturation. In *Proceedings of the American Control Conference*, Philadelphia, PA, 1998.

[13] R. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63, 1993.

[14] Y. Hu. Treatment learning: Implementation and application, 2003. Masters Thesis, Department of Electrical Engineering, University of British Columbia.

[15] G. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence Montreal, Quebec: Morgan Kaufmann*, pages 338–345, 1995. Available from `http://citeseer.ist.psu.edu/john95estimating.html`.

[16] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[17] R. Lutz and R. Woodhouse. Bi-directional analysis for certification of safety-critical software. In *1st International Software Assurance Certification Conference (ISACC'99)*, 1999. Available from `http://www.cs.iastate.edu/~rlutz/publications/isacc99.ps`.

[18] T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. Kiper. Condensing uncertainty via incremental treatment learning. In T. M. Khoshgoftaar, editor, *Software Engineering with Computational Intelligence*. Kluwer, 2003. Available from `http://menzies.us/pdf/02itar2.pdf`.

[19] T. Menzies and P. Compton. Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models. *Artificial Intelligence in Medicine*, 10:145–175, 1997. Available from `http://menzies.us/pdf/96aim.pdf`.

[20] T. Menzies and Y. Hu. Data mining for very busy people. In *IEEE Computer*, November 2003. Available from `http://menzies.us/pdf/03tar2.pdf`.

[21] T. Menzies and Y. Hu. Just enough learning (of association rules): The TAR2 treatment learner. In *Artificial Intelligence Review (to appear)*, 2004. Available from `http://menzies.us/pdf/02tar2.pdf`.

[22] T. Menzies and H. Singh. Many maybes mean (mostly) the same thing. In M. Madravio, editor, *Soft Computing in Software Engineering*. Springer-Verlag, 2003. Available from `http://menzies.us/pdf/03maybe.pdf`.

[23] N. Muscettola, P. P. Nayak, B. Pell, and B. Williams. Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence*, 103(1-2):5–48, August 1998.

[24] J. Perkins, A. Greenberg, J. Sharp, D. Cassard, and B. Massey. Free software and high-power rocketry: The portland state aerospace society. In *Proceedings of 2003 Usenix ATC, Freenix Track*, pages 245–258, 2003.

[25] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533 – 536, 1986.

[26] R. Rymon. An SE-tree based characterization of the induction problem. In *International Conference on Machine Learning*, pages 268–275, 1993.

[27] R. Rymon. An se-tree-based prime implicant generation algorithm. In *Annals of Math. and A.I., special issue on Model-Based Diagnosis*, volume 11, 1994. Available from `http://citeseer.nj.nec.com/193704.html`.

[28] S. Singh, S. Roy, and F. Rad. Itr: Rescuenet - embedded in-building sensor network to assist disaster rescue. NSF ITR Grant Number: 325014.

[29] Staff. Air transport market: The demand continues. *Aviation Today*, Aug. 2004.

[30] F. Winter. *Prelude to the Space Age: The Rocket Societies 1924-1940*. Smithsonian Institution Press, 1983.

[31] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[32] Y. Yang and G. I. Webb. A comparative study of discretization methods for naive-bayes classifiers. In *Proceedings of PKAW 2002: The 2002 Pacific Rim Knowledge Acquisition Workshop*, pages 159–173, 2002.

[33] Z. Z. Zheng and G. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41(1):53–84, 2000. Available from `http://www.csse.monash.edu/~webb/Files/ZhengWebb00.pdf`.

Part 6: Key Personnel and Bibliography of Directly Related Work

6.1 Eligibility of Key Personnel

This proposal comes from `bart-massey.com` LLC, DBA Uplift Systems, Inc.. Uplift Systems, Inc. is a Portland-based think tank doing advanced AI and open source research and development on a consultative basis for American companies. This is a small business concern as defined in section 2.14 of the solicitation document for this SBIR (i.e. organized for profit; American owned and controlled; operating within the USA; pays taxes to the U.S. state and federal government; has less than 500 employees).

The Principal Investigator, Dr. Tim Menzies, is Chief Information Officer at Uplift Systems, Inc.. Dr. Bart Massey is the founder, CEO, and owner of Uplift Systems, Inc..

6.2 Tim Menzies, Ph.D. (Principal Investigator)

Qualifications: Ph.D. AI, U. New South Wales 1995; M.Sc. Cog. Sci., UNSW 1988; B.Sc. CS., UNSW 1985.

- Currently Chief Information Officer at Uplift Systems, Inc., and a Research Associate Professor[4] of Computer Science at Portland State University.
- Author, over 150 refereed journal/conference/workshop articles.
- Former SE research chair, NASA software IV&V facility 2002–2004.
- Member, numerous IEEE conference program committees including ASE 2002–2004.
- Member, editorial board, Journal of Visual Computing and Languages.
- Reviewer, international refereed journals including CACM, IJHCS, Informaticia, IEEE Transactions on Software Engineering. Journal of Logic Programming, IEEE Software.
- Guest editor, special issues for IEEE Intelligent Systems; International Journal of Human and Computer Studies; Requirements Engineering Journal.
- Has run tutorials and workshops at international venues such as International Conference on Software Engineering (2000); Automates Software Engineering (2001); American Association for Artificial Intelligence (2002)

Eligibility: Dr. Menzies currently derives his income from active grants and from one day per week of teaching at Portland State University. He derives an intermittent additional income from his role at Uplift Systems, Inc..

During the period of performance for this project Menzies will suspend work on his other grants, and devote 100% of his time to this project.

As this work moves to Phase II, Uplift Systems, Inc. will hire more staff to expedite the work. For Phase II, Menzies will devote 60% of his time to this project.

Capabilities: Skilled in data mining methods; inventor of treatment learning [20]; extensive history in data mining and practical software engineering applications.

Basis for PI Selection: Apart from inventing treatment learning, Dr. Menzies has an excellent track record of successful work with NASA on numerous grants:

- The research rover, 2004: A NASA WVU USIP: $48,000
- See more! Learn more! Tell more!, 2003-2004: A NASA WVU USIP: $92,000
- A next-generation testable language, 2004: A NASA WVU USIP: $35,000
- Integrating model checking & procedural languages, 2003: A NASA IV&V DDF. $50,000 USA
- Understanding models better, 2003: A NASA WVU USIP: $17,000

[4] Research professors receive no fixed income; rather, their salary comes from any grants they bring in. As such, they have much flexibility in the organization of their work and no fixed time commitments to their University.

- A spectrum of IV&V techniques, 2002/2003: A NASA WVU CSIP. $200,000
- Better risk modelling, 2002: A NASA CSIP. $27,000
- Tree query languages, 2001: A NASA CSIP. $27,000

Bibliography of Directly Related Work: Menzies has evolved treatment learning over several years and several applications:

[1] E. Chiang and T. Menzies. Simulations for very early life cycle quality evaluations. *Software Process: Improvement and Practice*, 7(3-4):141–159, 2003. Available from `http://menzies.us/pdf/03spip.pdf`.

[2] M. Feather and T. Menzies. Converging on the optimal attainment of requirements. In *IEEE Joint Conference On Requirements Engineering ICRE'02 and RE'02, 9-13th September, University of Essen, Germany*, 2002. Available from `http://menzies.us/pdf/02re02.pdf`.

[3] T. Menzies. Practical machine learning for software engineering and knowledge engineering. In *Handbook of Software Engineering and Knowledge Engineering*. World-Scientific, December 2001. Available from `http://menzies.us/pdf/00ml.pdf`.

[4] T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. Kiper. Condensing uncertainty via incremental treatment learning. In T. M. Khoshgoftaar, editor, *Software Engineering with Computational Intelligence*. Kluwer, 2003. Available from `http://menzies.us/pdf/02itar2.pdf`.

[5] T. Menzies, R. Gunnalan, K. Appukutty, S. A, and Y. Hu. Learning tiny theories. In *International Journal on Artificial Intelligence Tools (IJAIT), to appear*, 2003. Available from `http://menzies.us/pdf/03select.pdf`.

[6] T. Menzies and Y. Hu. Constraining discussions in requirements engineering. In *First International Workshop on Model-based Requirements Engineering*, 2001. Available from `http://menzies.us/pdf/01lesstalk.pdf`.

[7] T. Menzies and Y. Hu. Reusing models for requirements engineering. In *First International Workshop on Model-based Requirements Engineering*, 2001. Available from `http://menzies.us/pdf/01reusere.pdf`.

[8] T. Menzies and Y. Hu. Data mining for very busy people. In *IEEE Computer*, November 2003. Available from `http://menzies.us/pdf/03tar2.pdf`.

[9] T. Menzies and Y. Hu. Just enough learning (of association rules): The TAR2 treatment learner. In *Artificial Intelligence Review (to appear)*, 2004. Available from `http://menzies.us/pdf/02tar2.pdf`.

[10] T. Menzies and J. Kiper. Better reasoning about software engineering activities. In *ASE-2001*, 2001. Available from `http://menzies.us/pdf/01ase.pdf`.

[11] T. Menzies, D. Raffo, S. on Setamanit, Y. Hu, and S. Tootoonian. Model-based tests of truisms. In *Proceedings of IEEE ASE 2002*, 2002. Available from `http://menzies.us/pdf/02truisms.pdf`.

[12] T. Menzies and H. Singh. Many maybes mean (mostly) the same thing. In M. Madravio, editor, *Soft Computing in Software Engineering*. Springer-Verlag, 2003. Available from `http://menzies.us/pdf/03maybe.pdf`.

[13] T. Menzies and E. Sinsel. Practical large scale what-if queries: Case studies with software risk assessment. In *Proceedings ASE 2000*, 2000. Available from `http://menzies.us/pdf/00ase.pdf`.

6.3 Bart Massey, Ph.D.

Qualifications: Ph.D. AI, U. Oregon CIRL 1999; M.Sc. CS, U. Oregon 1992; B.A. Physics, Reed College 1987.

- Founder, CEO, and owner of Uplift Systems, Inc.. Assistant Professor of Computer Science at Portland State University (9-month appointment). Instructor and Faculty Member, Oregon Master of Software Engineering Program.

- Lead Faculty Advisor, Portland State Aerospace Society. Principal architect of current avionics software and systems.
- Author, over 30 refereed journal/conference/workshop and trade press articles.
- Member, Usenix Annual Technical Conference (ATC) Freenix Program Committee 2003. Co-chair, Usenix ATC Freenix Program Committee 2004. Embedded Linux "Guru", Usenix ATC 2004. X Window System "Guru", Usenix ATC 2004.
- Reviewer, international refereed journals and conferences including National Conference on Artificial Intelligence, Software Quality Journal, IEEE Software.
- Architect and implementer of numerous and substantial open source software systems, including the XCB/XCL X binding library (`http://xcb.freedesktop.org`) and the Nickle Programming Language (`http://nickle.org`).

Eligibility: The period of performance for the project (June to September 2005) is the Summer term of PSU. Dr. Massey will not teach for that term. During the period of performance for this project Dr. Massey will devote 100% of his time to this project (June through September).

Capabilities: Degree work involved combinatorial search for general purpose planning and scheduling; experienced software engineer and SE instructor, specializing in applications of formal models and tools for evaluation; experience with machine learning in classification; 5 years experience with LV2 avionics systems; ; experienced programming language designer and implementer.

Bibliography of Directly Related Work: Massey has relevant publications in avionics, machine learning, and formal methods:

[1] James Perkins, Andrew Greenberg, Jamey Sharp, David Cassard, and Bart Massey. Free software and high-power rocketry: The Portland State Aerospace Society. In *Proc. 2003 Usenix Annual Technical Conference, Freenix Track*, San Antonio, TX, June 2003. URL `http://psas.pdx.edu/psas/usenix_2003/psas.pdf`.

[2] Bart Massey, Mick Thomure, Raya Budrevich, and Scott Long. Learning Spam: Simple techniques for freely-available software. In *Proc. 2003 Usenix Annual Technical Conference, Freenix Track*, San Antonio, TX, June 2003. URL `http://nexp.cs.pdx.edu/twiki-psam/pub/PSAM/PsamDocumentation/spam.pdf`.

[3] Bart Massey and Robert Bauer. X meets Z: Verifying correctness in the presence of POSIX Threads. In *Proc. 2002 Usenix Annual Technical Conference, Freenix Track*, Monterey, CA, June 2002. URL `http://freedesktop.org/Software/xcb/usenix-zxcb.pdf`.

Part 7: Relationship with Phase II or Future R/R&D

The goal of Phase II is to continue the research and validate the results of Phase I through actual test launches of the PSAS LV2 sounding rocket.

Recent PSAS launches have been at one of two ranges. A low-altitude range near Millikan, Oregon is only several hours from the PSAS home location in Portland, Oregon: this has made the Millikan site a convenient launch point for low-altitude ($20,000'$ ceiling) test launches. Mid-altitude launches ($100,000'$ ceiling) have been conducted in the Black Rock Desert of Nevada in conjunction with the Arizona High-Power Rocketry Association's annual international launch event. Exploration of additional launch site possibilities is part of the ongoing PSAS effort to combine convenient local access with the isolation needed for approval of higher-altitude launches.

Current PSAS rocket avionics include advanced flight data collection facilities. High-bandwidth instrumentation and operational data is collected on flash disk and recovered with the rocket. Most data is also transmitted over an 802.11b telemetry link and displayed in real-time using custom software. A seperate Amateur Television feed with data overlay capability is used as a backup downlink. This combination of redundant systems ensures a high likelihood of data recovery in the event of systems failure during flight.

In Phase II of this project, our AIVHMS will be integrated with the real-time navigation and control software of the LV2 rocket. This integration is facilitated by the modular design of the rocket hardware and software, and by our close familiarity with these rocket systems.

Once correct operation of the integrated AIVHMS is verified, additional launches will be performed to test the adaptive properties of Magic Bullet. Error conditions deemed recoverable in simulation will be deliberately induced in the rocket during flight. Given the nature of this sort of activity, numerous such launches are anticipated. Phase II will cost more than Phase I: it is anticipated that some launch vehicles will be damaged beyond repair during Magic Bullet AIVHMS shakedown. This additional cost is mitigated to some degree by the low parts cost of LV2: less than $5,000 total for the current airframe and avionics. However, PSAS may have to establish less labor-intensive rocket construction processes to meet demand: this may drive the price up slightly.

It is expected that the Phase II work will require additional investment in facilities and infrastructure, with costs to be borne solely by Uplift Systems, Inc.. Phase II will also require funds for travel to launch sites for launch activities.

Part 8: Company Information and Facilities

A sole proprietorship established in the mid-1990s, `bart-massey.com` LLC was organized as a Limited Liability Company in the State of Oregon in August 2004. Accounting services are provided by the firm of John S. Burles, CPA. The company currently does business as Uplift Systems, Inc. for the purposes of scientific research and development. Historically, `bart-massey.com` LLC has concentrated on consulting work involving software and computer systems technologies. Current and past clients include intellectual property law firms Klarquist, Sparkman LLP and Tonkon, Torp LLP, software services firm Critical Path, Inc., and international market research firm Sorensen Associates, Inc.

Current corporate technical infrastructure comprises the computers, software, and hardware tools needed to conduct the software simulation work required for the Phase I of this project. A network of several Linux-based computing systems is available for software research and development work. This includes a firewall machine and an auxilliary desktop machine with processors in the 1GHz range, running the Linux operating system. Ancillary computing and network equipment includes an 802.11b wireless network which can be properly secured as needed, laser and color inkjet printers, a high-resolution SCSI scanner, and a DSL modem. The current DSL network connection is 640Kbps download: this capacity can be easily expanded given need. System power is protected by UPS units. Backup is via a combination of RAID disk storage, DAT tape, DVD-R/RW, and offsite backup to partner locations. Sufficient electronic test and measurement equipment is present for work on computing and avionics hardware and software systems as needed. A low-frequency oscilloscope and digital and analog multimeters are available for circuit and systems analysis. Current company facilities comprise a dedicated space adequate to technical work. The work proposed for Phase I of Magic Bullet is primarily intellectual in nature: no extension of those facilities should be required for its completion.

Uplift Systems, Inc. is committed to technical excellence. While we are a young company, our success level has been high and our prospects are bright. Acceptance of this research proposal would be a significant milestone for the company. In any case, we expect continued healthy commercial growth.

Part 9: Sub-contracts and Consultants (Incl. Signed Commitment Letters)

Uplift Systems, Inc. is not hiring consultants or contractors in Phase I of this project. However, this letter of support from the Portland State Aerospace Society indicates the participation of PSAS in the proposed work.

Andrew Greenberg
Portland State Aerospace Society
Portland State University
PO Box 751
Portland, OR 97202–0751

September 7, 2004

Bart Massey
Proprietor
Uplift Systems, Inc.
17757 Schalit Way
Lake Oswego, OR 97035–5441

Prof Massey:

I am writing on behalf of the Portland State Aerospace Society (PSAS) to express our support for your proposed work with PSAS on the Magic Bullet Adaptive Intelligent Vehicle Health Management (AIVHM) system. I have carefully read your proposal, and it looks quite promising: we look forward to the opportunity to work cooperatively with you on this exciting technology.

We believe that the Magic Bullet AIVHMS will be valuable in supporting our transition into fully guided and managed flight. We also believe that Uplift Systems, Inc. is an ideal partner for this transition. You have been the PSAS PSU lead faculty advisor for some time. You understand our LV2 sounding rocket well and have contributed to its avionics system design in the past. Our vision statement is to inexpensively put microsatellites into orbit. We know that doing so will require designing and implementing advanced avionics for navigation, control, and VHM. We believe that the Magic Bullet AIVHMS will be quite useful in this quest. PSAS is prepared to provide the necessary technical support and cooperation to make the Magic Bullet AIVHMS project succeed.

Please let me know if there is any way we can be of assistance to you in this matter. Thank you for your attention!

Sincerely,

Andrew Greenberg
Project Manager
Portland State Aerospace Society

Part 10: Commercial Potential Applications

Widely available hardware is rapidly becoming more capable and sophisticated. The bottleneck in using this hardware is the effort and complexity involved in creating the software controllers for it. We believe that automatic data mining will significantly reduce the systems engineering cost of such controllers.

Nowhere would this be more useful than in the avionics industry. The size of the commercial avionics market is staggering. The precise size is unknown: however, one estimate placed that market at $4–5 billion per year [29]. A significant factor slowing the growth of that industry is the cost of software. Over half the budget of the latest Boeing 777 was spent on software development.

Figure 9: SpaceShipOne.

Airframe and avionics hardware capabilities that previously required the resources of a major goverment to achieve are now within the reach of industrial and even amateur groups. In 2004, for example, the privately funded manned SpaceShipOne (Figure 9) flew to 100 kilometers (62 miles) in altitude, leaving the Earth's atmosphere. The PSAS LV2 airframe is hardly as impressive as SpaceShipOne. Nevertheless, PSAS has produce performance equivalents to standard rocket avionics hardware systems at a fraction of the standard commercial cost:

- The PSAS 6-axis Inertial Measurement Unit can be built with top-quality MEMS components for a few hundred dollars, and achieves performance similar to low-end commercial fiber ring gyros and linear accelerometers costing up to 100 times as much.
- The current PSAS medium-altitude telemetry down link is based on COTS 802.11b technologies: again, a few hundred dollars worth of components do the job of a system that might cost tens or even hundreds of thousands of dollars if purchased from commercial vendors.

Despite these advances in low-cost hardware systems, the software engineering effort required to produce a quality navigation and control system still represents a major investment in time and money. A recent highly unsystematic estimate of the commercial cost of development of the current PSAS software base (using SLOC with a COCOMO model) suggests that the software component of PSAS systems might require 6 commercial developers one calendar year to replicate, with a cost in the neighborhood of $1M. This cost is disproportionate to the cost of the hardware and airframe. Worse, it is well understood in the software engineering community that such a large and complex software infrastructure represents a significant risk of serious undiagnosed defects, even when best practices are followed. With loss of airframe a common consequence of sounding rocket software failure, and given the slight risk of actual human injury from these failures, it is highly desirable to do better.

Automated systems engineering using (e.g.) The Magic Bullet AIVHMS therefore represent the breakthrough technology required to reach the goal of ultra-low-cost avionics. Critical components of the PSAS avionics system, for example, will be significantly simplified by Magic Bullet. More importantly, Magic Bullet's ability to adapt in the presence of failure will significantly mitigate the risk of failures in hardware, airframe, and even ancillary software systems.

With the addition of Magic Bullet, Uplift Systems, Inc. should be well-poised to sell commercial avionics systems and design services into a variety of organizations. Obviously, sounding rockets within NASA and elsewhere are prime candidates for the Magic Bullet AIVHMS. Groups exploring Unmanned Aerial Vehicle avionics (including NASA) should also find AIVHM useful: indeed, almost any kind of unmanned autonomous vehicle, including land and underwater craft, should be able to benefit from the Magic Bullet AIVHMS. While it may be difficult to safety-qualify the Magic Bullet AIVHMS as a primary controller for human flight in the short term, it should nonetheless be usable in controlled-responsibility ancillary systems for commercial avionics.

In the longer term, projects where the success of the mission relies on deploying a large number of low-cost robots might use Magic Bullet technology to maintain the health of the ensemble. Examples of such applications might include NASA SWARM missions, or civil defense rescue missions in crushed buildings (c.f. the work of Portland State University CS Prof. Suresh Singh and others [28]) or other hostile environments.

Part 11: Similar Proposals and Awards

Menzies is currently working on several proposals involved with data mining:

	agency					PI	
id	name	address	performance dates	topic	title	name	title
i.	NASA SARP	100 University Ave, Fairmont, WV, 26554	2004-2006	meta-heuristics, data mining, modeling	Next generation testable languages	Tim Menzies	Dr.
ii.	NASA SARP	100 University Ave, Fairmont, WV, 26554	2003-2005	data mining, defect detectors	See more Learn More Tell More	Tim Menzies	Dr.

Those projects are almost entirely unrelated to the proposed work. Project (i) explores non-real-time applications of treatment learning for ground systems. Project (ii) is unrelated to treatment learning.