JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, August 2005

Unmanned Aerial Vehicles: Autonomous Control Challenges, a Researcher's Perspective

Bruce T. Clough[*]
Air Force Research Laboratory Wright-Patterson AFB, OH

Three years ago we outlined the basic challenges faced by autonomous control researchers as intelligence, safety and affordability:

- **Build intelligence so that the UAVs can do tasks which would normally take a manned aircraft with minimal human oversight.**
- **Instill safety such that we will trust and want to use the systems.**
- **Make the systems affordable alternatives to manned aircraft, because even the most wonderful technology is of no use if we can't pay for it.**

Now it's three years later, and quite a few million dollars spent, so how are we doing? Although we don't have the final answers, we know a lot more about the questions we should be asking. We also know a lot more about where we should be spending our money to enable UAV operations—the "we needs":

- **We need to instill situational awareness in our UAVs, especially in the area of airspace operations.**
- **We need to ensure that the autonomous UAVs and their human supervisors understand each other's intent and actions.**
- **We need to build the proper mental models inside the UAVs "brain" such that they plan and react as we (humans) would, yet allowing the UAVs some level of discretion on task execution such that they (at least for military aircraft) are not 100% predictable.**
- **We need to advance software methodologies to test huge blocks of safety (and flight) critical code.**
- **We need to keep exploring the "realm of the possible" to see which paradigms we are clinging to require casting off.**
- **We need to make our autonomy flexible.**

These points, as well as others, and lessons learned from our autonomous control programs, are examined in this paper—developing our research vision and direction over the next three years—outlining the areas of interest and what we expect to get from them. Will we make our goals? Don't know—wait three years and look for the next report card.

I. Introduction

THE Air Force Research Laboratory is dedicated to delivering the latest technology to ensure the cutting edge of the joint warfighter is sharp. One of our current focuses is developing command and control systems to increase the utility, performance, and safety of unmanned aerial vehicles (UAVs). Over the last ten years technologies of many types have made the transition from lab to battlefield, with the result being UAVs are relied on for more and more tasks, from reconnoitering the battlefield to actually dropping weapons. In the future they will do more—much more. Command and control is one of the most important functions for UAVs, especially if they carry deadly force. This paper discusses the lessons learned facing the challenge of developing autonomous control—the capability for UAVs

Received 3 October 2003; revision received 25 October 2004; accepted for publication 18 October 2004.

[*] Chief, Future Capability Branch. Bruce.Clough@wpafb.af.mil

to make their own decisions. Autonomous control enables many functions, normally accomplished by humans, to be done by the vehicles instead, allowing the human operator to command multiple UAVs, both increasing the type of missions UAVs can accomplish (performance increase) and reducing the number of humans required to operate UAVs (affordability). This paper revisits several challenges laid out in a paper three years ago,[1] discussing how far we've come, how far yet we have to go, and how we plan on getting there.

II. Background

Three years ago this author outlined the three main challenges faced by developers of UAV autonomy, the ability for UAVs to make decisions on their own. As mentioned, this ability both increases the utility of the UAVs as well the span of control for the UAV operator. The goal is to enable safe/reliable autonomy, changing the UAV operator into a UAV supervisor. We humans should be able to trust the UAVs to accomplish a mission, and enable them—giving them tasks and goals, rather than orchestrating their every move. This would allow the UAV operator to spend more time working on the "big picture" rather than the detailed tasks flying an aircraft and plotting every pathway through the sky.

Our definition of autonomy is simply the capability for the UAV to make a decision. What types of decisions we allow the UAV to make, their breadth and impact, relates to how autonomous the UAV is—more on that later.[2]

In the process of developing technology to increase UAV autonomy we ran headlong into many challenges, the topic of the original paper.[1] Now three years later, and many dollars spent, we have amassed many lessons learned on the way to overcoming the challenges. Have we solved all of them? No. Are we making progress? Yes.

The rest of this paper lays out the original challenges, and then looks at the progress we made towards them. We examine what we've found out, and what is still yet to find. Wrapping up the conclusions section is a list of questions we hope to be able to answer in the next three years.

III. The Challenges

Basically, we can put the autonomous control challenges faced by Air Force researchers into three buckets: building intelligence, instilling safety, and enabling affordability.

- Building intelligence—to do a task a human does require some reasoning capability. Obviously, for some tasks the full reasoning capability of a human is needed. For others, maybe not. Autonomous UAVs require the intelligence required to accomplish the task(s).
- Instilling safety—autonomy without trust will not be used. We must ensure the UAVs will follow our directions, and moreover, not do anything "stupid", either by their own doings or by human commands [*Note 1*].
- Enabling affordability—if we can't afford it, we won't use it. Adding functionality never comes cheap. We have to develop our UAV autonomy such that the intelligent and safe system also results in a total ownership cost per capability that will not break the bank.[3]

Let's examine each of these autonomy challenges separately, and the lessons we've learned so far in each:

A. Making Them Think On Their Own

Giving UAVs some sort of reasoning capability is the first step on the long road to intelligent autonomy. Given that the goal is to take the human out of the cockpit, yet leave the human's capability in the UAV, this is a serious challenge. However, before we marched off in building intelligent autonomy, we first had to establish a metric by which we could measure our progress. The Autonomous Control Level (ACL) was that metric, but the key to developing it was thinking "human".[2]

1. Developing An Autonomy Metric

The ACL was ground breaking in the sense that it related the impact of the decisions a UAV could make to a number—thus denoting a level. Prior autonomy metrics tended to relate to who made the decision (Human, UAV and Human mixed, UAV alone), not what the decision was, or its impact. Thus those earlier metrics were not much help answering the "how autonomous is our UAV?" question. We now had a way of relating human decision to machine decision capability. This capability continuum is shown in Fig. 1 versus the ACL level.

But even at this level the march from ACL level to ACL level was a bit arbitrary, so to make it easier to measure attributes of the autonomy, it was split up into four sections, based on how humans make decisions—John Boyd's

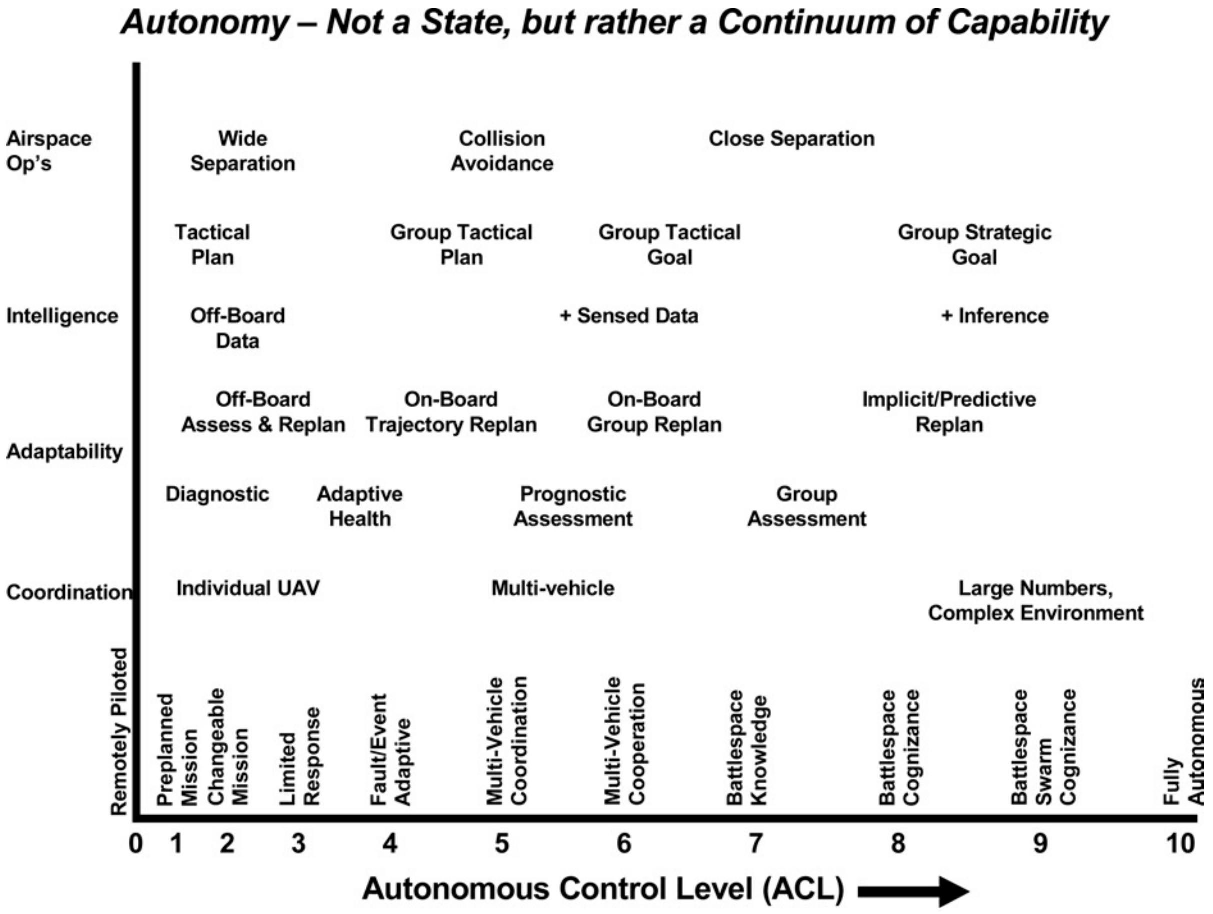


Fig. 1 Machine decision capabilities versus ACL level.

OODA (observe, orient, decide, act) loop.[4] Doing this allowed us finer measurement at what level UAV decisions are made, and to target specific gaps in technology with R&D efforts.

The ACL has been used to measure our own research programs, plan new programs, and compare autonomous control technology efforts. Cited in the latest DOD UAV Roadmap,[5] versions of the ACL are being tailored for other autonomous vehicles in other environments.

Our investment in future autonomy development is tied to our ACL goals. We plan on reaching an ACL 6 at a NASA technical readiness level (TRL) of 6 by 2007, and ACL 8 at TRL 6 by 2013. TRL 6 is the point in technology development where the technology is ready to be transferred from the Laboratory to our customers in a System Program Office (SPO) for Engineering, Manufacturing and Development (EM&D). Our technology roadmaps reflect this in the programs planned. Figure 2 is a graphic reflecting this.

Since the steps between ACL levels are exponential in difficulty, these are ambitious goals; however, we are well on our way of making good the first goal, and our plans point the way to the second.

2. *Lesson Learned: The Hardest Part Of Making A Decision Isn't Deciding, It's Knowing What To Decide With*

Controls folks spend most of their time with autonomous systems figuring out the best way of making a decision. Should we optimize? Should we be robust? Should we treat our software algorithms as artificial intelligence agents? What does the cost function look like?

We're missing the boat. The hardest part about making a decision is figuring out what we're deciding about. What is the situation? If we would spend some time with our human effectiveness brethren we'd know this. We would know all about situational awareness.

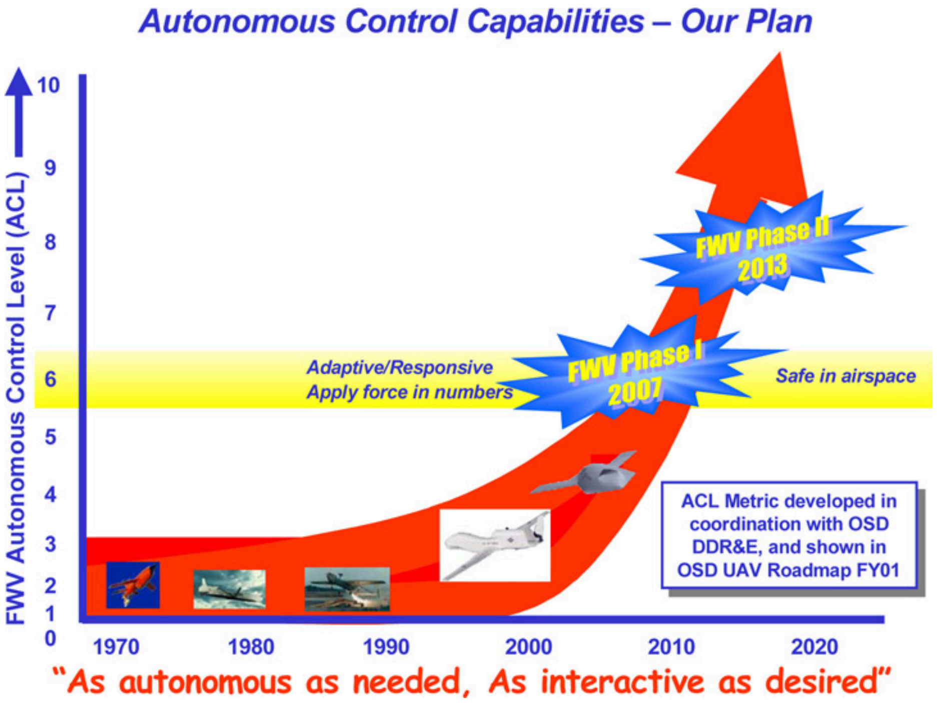


Fig. 2 Autonomy level plans for UAV R&D.

To make good decisions, humans have long sought after better sources of information to make decisions with. Autonomous UAVs are no different. As stated above, our autonomy metrics are based on Boyd's OODA loop, describing human decision-making in order to force us to consider how information arrives, what it means, what we do about it, and how we carry it out.

If one looks at what it takes to do each of these parts of the OODA loop, one finds that deciding is the easy part. Figuring out what to base our decision on is the hard part, the part that takes true intelligence. We don't want to know there are a lot of tanks out there; we want to know that we're being attacked by an armored division.

Thus orienting is critical. Most existing systems assume the human is doing the orienting—they ensure that the human part of the system gets the right information at the right time to support decisions. If we are to increase UAV autonomy, develop teams of UAVs, then we must develop the ability to orient in the UAVs, we must give them some level of intelligence.

3. Lesson Learned: "Best" Autonomy Method Used Is Related To Task To Accomplish—There Is No Optimal Method For Any Task

Just as the ACL had to be linked to the task a UAV was doing in order to define a meaningful level, the task a UAV has to accomplish is linked to the "best" autonomy development method family to "autonomate" it. Whether it be optimizing, bio-inspired, control-theoretic, artificial intelligence rule based, or some cognitive science construct each has its place in the autonomy developers' tool box. Reference 6 outlines an approach to narrow down what method to use on a specific task based on attributes of the task and autonomy development method. The main attributes came down to four:

- Planning Requirements—Reactive or Deliberate?
- Information Requirements—Local or Global?
- Location (of processing)—Central or Distributed?
- Rule Construction—Functional or Semantic?

Comparing these attributes between the task and autonomy building methods is a first test for choosing a decent method to accomplish the task. Like attributes amongst task and method indicates methods to try first.

But note the above is just a guide. Any autonomy development method can be used for any task, it's just that some will give workable answers in the real world faster than others, and faster development time relates to a more affordable design and development effort. This is the second part to this lesson learned: there is no perfect method, use what works, and what makes sense.

4. *Result: Swarming (Emergent Behavior) Is Useful in The UAV World*

As the above indicates, methods using emergent behavior (reactive decentralized, local, semantic—or RDLS) should be useful for something, not just a joke. But what are those tasks? Taking a cue from above, we related the task to autonomy, and developed a list of six tasks emergent behavior (in our definition, swarming is a type of emergent behavior) is useful for.[7] These are:

- Area search and attack—the ability to find targets of interest in an area of interest.
- Surveillance and suppression—the ability to loiter and attack targets of opportunity.
- Diversion—the ability to slow down decision cycles and tie up the opponents resources.
- Psychological warfare—leverage the sinister human connotation with "swarms" for controlling crowds.
- Survivability—leverage the common tendency of swarming entities for probabilistic motion to reduce chances of being hit.
- Software reduction—rely on emergence to reduce the software lines of code required to achieve an effect.

Of the above, our main focus is software reduction. As will be reported later in this paper, the advent of autonomy results in significant code increases on the vehicle, with the attending costs of code development and maintenance. We have demonstrated software algorithms using emergent behavior for formation flight and other attack tasks. In one effort,[8] complicated strike tasks were accomplished using a few pages of code programmed into each of the UAVs—code that could run real-time on a 486 processor that also happened to be running the rest of the functions of the vehicle.

5. *Result: Leader-Follower Multi-UAV Software Ready To Transition To Flight Tests*

Via the Integrated Tactical Aircraft Control (ITAC) program multi-UAV autonomous control technology is ready to enter demonstrations.[9] This effort has quietly been developing leader-follower, mimicking human command structures (centralized planning, decentralized execution with some responsibilities delegated to the followers) and simulating them in action. This effort assumes the human tasks a UAV team and they execute the mission (except for weapons release authorization) by themselves.

The ITAC software agent architecture is shown in Fig. 3. In each vehicle there are three levels of agents: package agents that generate tasks for the strike package, vehicle level agents that generate commands to vehicle subsystems in response to package tasking, and subsystem agents that ensure the vehicle can respond to package tasking.

Two of the more significant lesson learned coming from ITAC effort are:[10]

1. "*Centralized planning, distributed execution*" is a nice phrase, but leaves a lot of wiggle room. The ITAC experience shows that indeed, centralized planning simplifies the coordination of the actions of multiple vehicles. But it also constrains the flexibility and responsiveness of the vehicles as all actions must be cleared through the centralized authority in order to maintain coordination. The challenge is to push as much autonomy or decision-making capability down to the vehicles to allow them to operate with greater latitude but still coordinate actions and trajectories.
2. A good mission plan database can serve a variety of useful roles in an autonomous system. First it embodies the pre-mission planned view of the optimal mission plan. During the mission it is the means by which changes to the mission plan are shared amongst vehicles. It also provides each vehicle with a "model" of what each other vehicle is going to do, allowing vehicles to remain coordinated even during loss of communication by simply "flying the plan". Finally this mission plan database serves as a record of changes to the pre-mission plan, available for post-flight analysis. All this being said, experience shows that updating this database throughout the mission and sharing it amongst the vehicles can use significant data link bandwidth.

Although ITAC is now over with, the technology has been transitioned to other programs to mature to flight test as well as to be used in simulations to define the "realm of the possible" with this type of autonomy. Leader-follower has reached a technology readiness level (TRL) of 5 and is being prepped to go on to TRL 6.

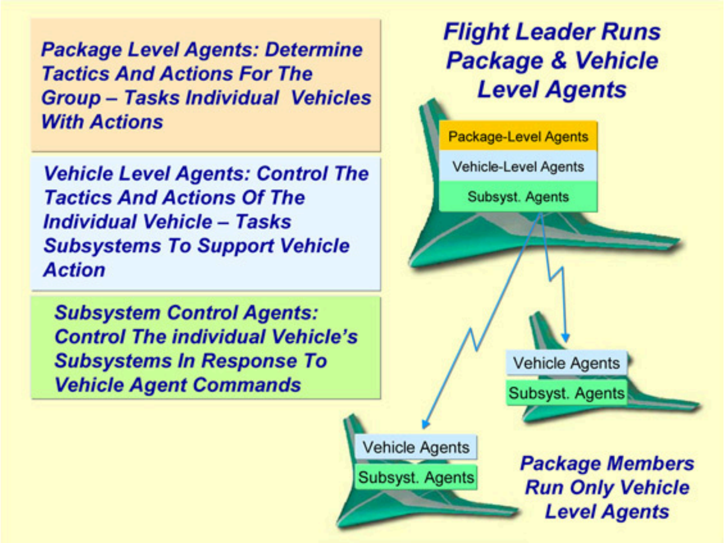

Package Level Agents: Determine Tactics And Actions For The Group – Tasks Individual Vehicles With Actions
Vehicle Level Agents: Control The Tactics And Actions Of The Individual Vehicle – Tasks Subsystems To Support Vehicle Action
Subsystem Control Agents: Control The individual Vehicle's Subsystems In Response To Vehicle Agent Commands
Flight Leader Runs Package & Vehicle Level Agents
Package-Level Agents
Vehicle-Level Agents
Subsyst. Agents
Vehicle Agents
Subsyst. Agents
Vehicle Agents
Subsyst. Agents
Package Members Run Only Vehicle Level Agents

Fig. 3 ITAC multi-UAV C2 architecture.

6. Result: Distributed Multi-UAV Autonomous Control Moving Into Engineering Development

Not just a theoretical curiosity anymore, the technology required to allow UAVs to come to a group agreement on the best way to accomplish tasks (rather than a leader-follower set-up as in ITAC) is being prepped for 6.3 entry. Programs such as the AFRL Distributed Cooperative Autonomous Teams (DCAT) and others are aimed at showing the purported benefits of distributing the decision-making ability of the UAV team including:

- Immunity to leader loss.
- Robust to member loss.
- Capability to handle partial data.

However, we also want to investigate possible drawbacks:

- Excess communications required for decisions.
- Least-common denominator decisions—no tough choices.
- Lack of common big picture reduces allowable decisions or drives high data rates.
- Some decision strategies non-deterministic.

Over the next 3–4 years we should develop the results to compare distributed group UAV decision making with the hierarchical UAV control scheme of ITAC. We will compare them side-by-side in the same simulation environments in the same scenarios. From this we should be able to determine when it is best to use which technique—getting back to the notion that there is not best technique, just better techniques to use for specific tasks.

7. Lesson Learned: Need To Build Conversations Between Autonomous UAVs and Humans

Not only must humans be able to talk to UAVs—we the UAV autonomy designers have to insure that the UAVs understand human intent—but also we need to make sure the human operator understands our intent. Understanding of intent is a key factor in developing trust. And UAVs will only get to make decisions if the humans trust the outcomes. For instance, the following conversation was heard on an O'Hare Airport Ground Frequency:

Controller: "American 443, proceed to runway . . ."
American 443: "American 443, ummm, we have a plane in front of us."
Controller: Okay, who is the American at . . .?"
American 135: "American 135"
Controller: "American 135, do you have anyone in front of you?"
American 135: "American 135, no"
Controller: "American 135, okay, you proceed to runway . . ."

We need to give UAVs this same ability to talk back. There might come a time where our UAVs, being situated in the environment, might know better than the operator on how to do something, but how do we get this across to the human? How does the UAV tell the human that the instruction it just received seemed stupid? Most aircraft crashes of large aircraft are due to human error, just taking the human out of the cockpit does not eliminate error chances.[3,11] We need to work on this since the correct man-machine interaction is crucial for effective teaming. As an example, several years ago an errant self-destruct signal caused the loss of a Global Hawk. What if instead of blindly following the command the UAV saw that it was doing the programmed mission in the programmed manner in good health and asked the human:

"*Gee human operator, do you really want me to self-destruct*?"

8. *Result: Established Inter-Governmental Agency Intelligent Autonomy Working Group*

Let's face it, there are a lot of different agencies across the Federal Government developing autonomous technologies. Within the DOD itself you have several directorates of the Air Force Research Laboratory, Naval Research Laboratory, Office of Naval Research, Army Applied Aviation Technology Directorate, Night Vision Laboratory, Aviation and Missile Research and Development Center, Joint Robotic Program, Navy SPAWAR, and a host of other organizations including numerous DARPA programs. All are looking at developing autonomy from different perspectives for different tasks. Wouldn't it be nice if these had a chance to learn from each other? There is direction from top (OSD DDR&E) to coordinate UAV research, but this doesn't cover undersea, surface, or ground vehicle technology development.

In the summer of 2001 research leads from ONR, AFRL, and AMRDEC concluded that a forum for discussing and cross-fertilization of autonomy developments at the working level needed establishment. September of 2001 saw the first meeting of the DOD Intelligent Autonomy Working Group. Since then it has been expanded to include other organizations working autonomy, including NASA and NIST. Our near term goal is to establish common definitions and metrics for measuring autonomy, and progress in building autonomy. Longer-term goals are to cross-link research and products across agencies.

B. Making Them Safe

UAVs are deemed unsafe since they are unreliable. Unreliable systems cannot be trusted. A review of the available crash data shows:[11]

- UAVs Crash At Rates Well Above Manned Aircraft.
- Unit Cost More Important Than Reliability.
- To Increase Trust We Must Increase Reliability/Predictability.

As Fig. 4 shows, this distrust is well founded. Current UAV unreliability is due to the lack of investment in areas that make them reliable. In other words, UAVs are unreliable because we want them to be inexpensive. As the figure shows, current UAVs, even expensive ones, are at least an order of magnitude less reliable than the worst fixed-wing manned aircraft. As shown in Fig. 5, developed from examining the causes of hundreds of UAV accidents, the reason for crashes are just as interesting.

Humans are the leading cause, mostly from lack of situational awareness while acting as remote pilots. One might expect automating those functions will eliminate a source of problems, which is a successful tack taken by a few programs.

In order to improve UAV safety, we first had to define how safe a UAV must be. Safety can be looked at in a lot of ways; the way we decided to split it up was safety due to performance and safety due to reliability:

- Performance—Nominal operation of the system will not cause an unsafe situation.
- Reliability—Failures within the system will not cause an unsafe situation for the system.

Trust when it comes to sensing has two parts: do I have the performance to "see" as far as I have to, and how reliable is my system? Both are equally important, but only the first is considered by most efforts. As an example, Fig. 6 shows the safety aspects of both in an effort giving UAVs "see & avoid" capability. Ignoring either is a mistake since the certification officials will be interested in both, and systems that cannot be certified cannot transition to the operational community. We have made the conscious effort to consider reliability in our development efforts, which is wise since the numbers show that any particular sensor doesn't have enough reliability.[12]

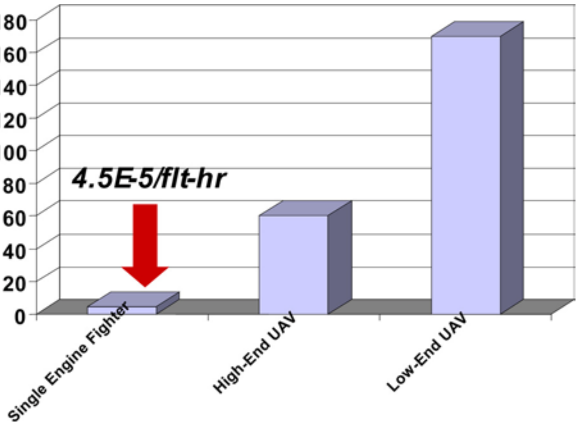


Fig. 4 Comparison of manned and unmanned aircraft loss rates (per 100 K Flt Hrs).

Improved safety from improved reliability will carry a price tag; however, it is the opinion of this author that the price tag is reasonable and the return immense.[3]

Safety must be measured in order for it to be improved. We had overall aircraft loss rates. The challenge was to derive metrics we could measure with, and design to.

1. Result: Redefined "Flight Critical" For UAVs

Most of what we flight control folks worry about is functions that, if they fail, will result in the loss of the aircraft and pilot. These are the so-called "Flight critical" functions. Flight critical is defined in terms of flying qualities:

> *A function is critical if the loss of the function results in an unsafe condition or inability to maintain FCS* [Flight Control System] *Operational State III. Op State III relates to minimum safe operation, the level of minimum functionality required for safe return and landing. In terms of flying qualities it satisfies at least Level 3.*[13]

In other words, if a failure happens and the pilot cannot keep the pointy-end forward, that was flight critical. However, by definition, no man is aboard a UAV, so the old idea of flight critical functions being related to flying qualities level needs to be revisited. At a minimum level, "flight critical" might be considered such that the loss of

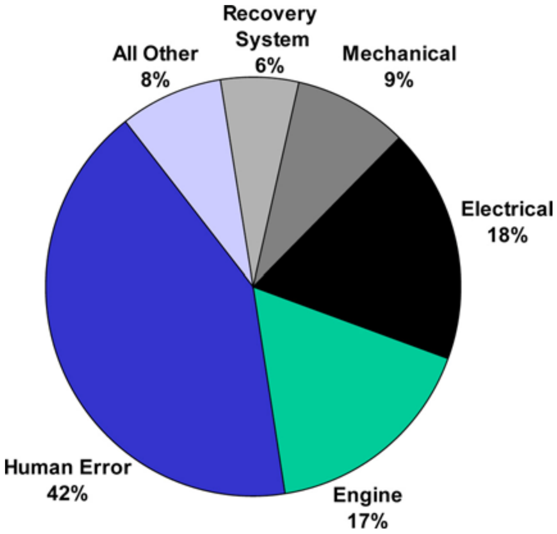
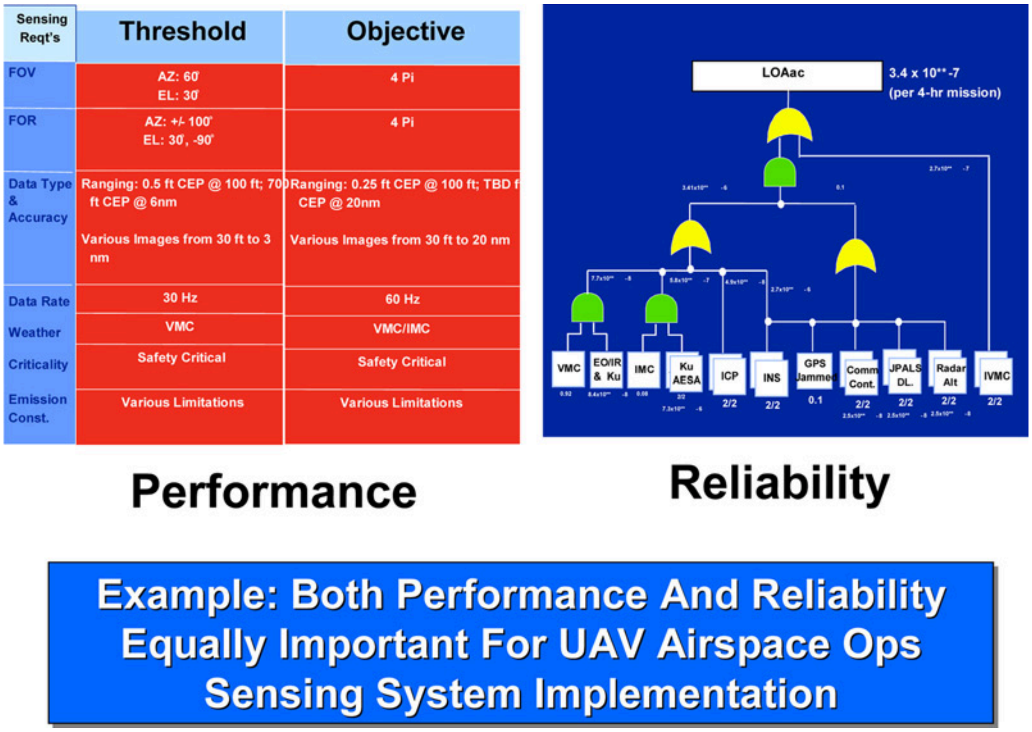


Fig. 5 Reasons for UAV loss.



Sensing Reqt's	Threshold	Objective
FOV	AZ: 60˚ EL: 30˚	4 Pi
FOR	AZ: +/- 100˚ EL: 30˚, -90˚	4 Pi
Data Type & Accuracy	Ranging: 0.5 ft CEP @ 100 ft; 700 ft CEP @ 6nm Various Images from 30 ft to 3 nm	Ranging: 0.25 ft CEP @ 100 ft; TBD f CEP @ 20nm Various Images from 30 ft to 20 nm
Data Rate	30 Hz	60 Hz
Weather	VMC	VMC/IMC
Criticality	Safety Critical	Safety Critical
Emission Const.	Various Limitations	Various Limitations

Performance

Reliability

Example: Both Performance And Reliability Equally Important For UAV Airspace Ops Sensing System Implementation

Fig. 6 Balance between performance & reliability for safety.

that function precludes a safe return to base, which is equivalent to a safe return by a pilot. That way no bystander will be injured by the UAV. For UAVs the injuries will be to third parties. But is that it? What about the on-board software that makes decisions impacting safety of other humans, such as dropping bombs? We've come up with another definition of flight critical for autonomous vehicles:

Flight Critical: *Ability not to harm friendlies*.

"Friendlies" is used since there might be times we want to hurt the "un-friendlies", and our system must know the difference. The astute reader will notice this is written in the same manner as Isaac Asimov's famous "Robotic Laws". We made a conscious decision to modify the Robotics' Laws[14] and apply them to our work to insure autonomous system safety. Our "UAV Rules" are:

- *First Law*: A UAV may not injure a friendly, or, through inaction, allow a friendly to come to harm.
- *Second Law*: A UAV must obey directions given by friendlies except where such orders would violate the First Law.
- *Third Law*: A UAV must protect its own existence as long as such protection does not conflict with the First and Second Laws.

The First Law is the safety requirement. All flight critical functions support the requirement not to harm friendlies, whether they are actuator monitors assuring equipment failures won't result in the UAV crashing into Aunt Bertha's house, the automatic collision avoidance algorithms insuring the UAV won't hit a manned aircraft, or possibly even the automatic target recognition algorithms mistaking a refugee bus as a T-72.

The Second Law can be looked at as eliminating "friendly fire". It is also the sanity check to avoid My Lai massacres and other tragedies.

The Third Law can be considered threat or obstacle avoidance (as long as the obstacle doesn't have friendlies on it, then see First Rule). As with the robotics law, the Third Law might result in UAVs sacrificing themselves to save friendlies.

We have used these rules in our autonomy development efforts. Programs such as Automatic Air Collision Avoidance System (Auto-ACAS) and Automated Aerial Refueling use these rules to develop conflict avoidance algorithms. These rules also helped us develop another needed requirement for on-board autonomy, its reliability.

2. Result: Defined Reliability As Part Of "As Safe As A Manned Aircraft" For Combat UAVs

So how reliable must a combat UAV be? This number is important since it drives our R&D expenditures developing reliable systems. From UAV failure rates[11] we know how reliable UAVs are, but no specific reliability targets exist in print. It was up to us to define it, and then ask for blessing. To define, we took the phrase "as safe as manned aircraft" literally, and examined why, and how, manned aircraft fail.[11] From this number we established what we need at a minimum to be as safe as a manned aircraft (threshold of $2.3 * 10^{-5}$ failures/flt-hr), a more ambitious number (objective of $1 * 10^{-6}$), and a desirability curve which joins the two and indicates how happy the customer is with the result [*Note 2*]. Figure 7 details the Loss of Aircraft (LOA) due to autonomous control (LOA_{ac}) desirability curve.

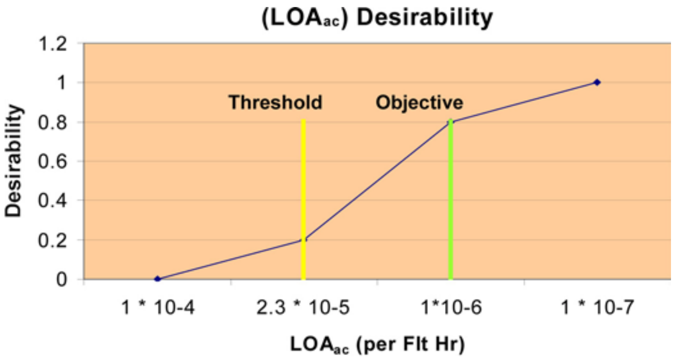
Defining this metric is a key enabler of UAV control system development since it defines redundancy levels, software partitioning and testing, component testing, etc. This AFRL metric has since been used to shape several control system development efforts, both inside and outside AFRL, and is now a DOD Level metric [*Note 3*].

3. Result: Defined New Areas Needing Flight/Safety Critical Analysis For Autonomous UAVs

Before autonomy, the control system designer took the loss rate (failure rate) number given to him/her by the requirements, then partitioned that out to the components, determining how good they had to be, how redundant, how well tested, etc. This can be looked at as divvying up the requirement like a pie. Well, autonomy brings a few new pieces to the pie that designers have to consider - these are shown in Fig. 8.

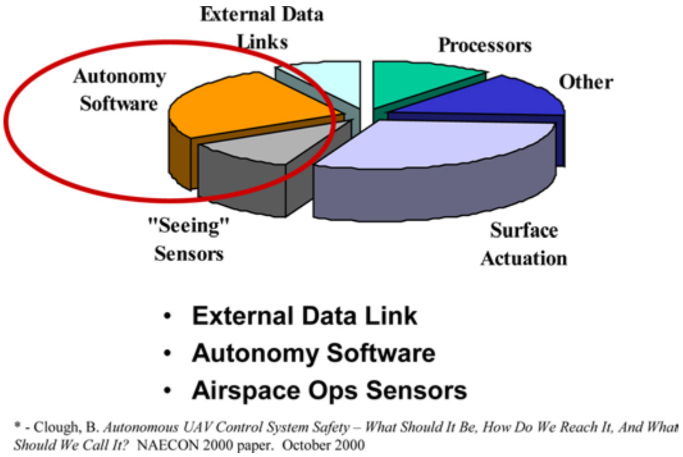
The new additions to the failure pie and the reasons for inclusion are:

- External Data Links—Intents and decisions of the UAVs will be based on information coming across data links. Bad decisions could result in people killed. Normally humans filter this information, and handle the situation when the communication links fail, now we either need more reliable communication links, or more intelligence to sort bad data and handle the information outage.
- Autonomy Software—The "pilot" aboard an autonomous UAV is the autonomy software, a software load that sits on top of the software normally in aircraft control systems. Since this software has the visibly important job of being the pilot, and since it also is an application in an open architecture (more on this later), it makes sense to call attention to it, rather than include it as a part of Processors.
- Airspace Operation Sensing—These are the sensors put on-board the UAV to replace the pilot's senses, especially when it comes to airspace operation requirements such as see & avoid, runway incursions, due regard, etc. Operations, which if not performed correctly, will lead to loss of life.



To Be As "Good As Manned Aircraft The Loss Rate Of Aircraft Due To Control System Failure (LOA_{ac}) Is LOA_{ac} = 0.46(4.5E-5) = 2.3E-5/flt-hr

Fig. 7 Defined loss of aircraft (LOA) rate for combat UAVs equivalent to manned aircraft.



- External Data Link
- Autonomy Software
- Airspace Ops Sensors

* - Clough, B. *Autonomous UAV Control System Safety – What Should It Be, How Do We Reach It, And What Should We Call It?* NAECON 2000 paper. October 2000

Fig. 8 New influences on aircraft loss due to the inclusion of autonomous control.

After recognizing these new players in the ‘Loss Of Aircraft’ metric, we put in place programs to investigate how best to mitigate the safety risk posed by them.

- The Verification and Validation of Intelligent Adaptive Control Systems (VVIACS) program is developing methodologies to certify autonomous control software.
- The Distributed and Infocentric Reliable Control Technology (DIReCT) effort is evaluating communication link effects and mitigation strategies for distributed control systems (such as a UAV team).
- The Autonomous Flight Control Sensing Technology (AFCST) program is evaluating sensor performance and reliability requirements for airspace ops sensing.

In each case, a need was noticed, and filled.

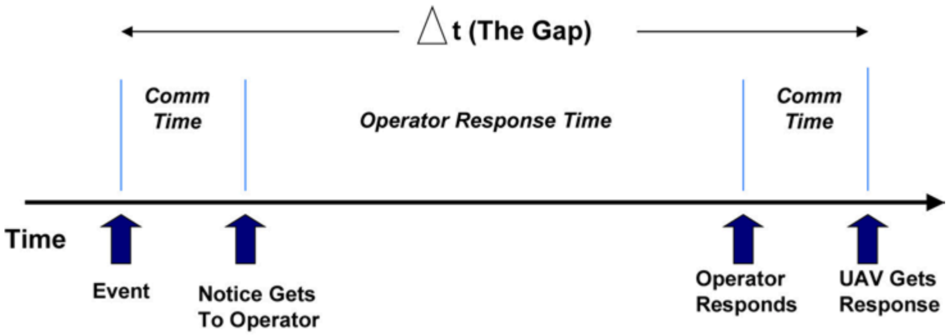
4. *Lesson Learned: Mind the Gap!*

Currently UAV autonomy designers assume that the UAV operator can instantly contact their UAV charges almost instantly anywhere in the world. Actually, this is not the case, and it drives what decision capability one has to put on board. It takes a finite transit time for information between UAV and UAV operator, possibly 7 seconds or more. If the operator is involved in a decision, either doing it, or monitoring it, it takes time for the human to understand the situation and to make the decision. Adding up the information transit and decision times means that it can be well over 30 seconds from when something happens to when the operator can effect a change in the UAV. We call that time “The Gap”, the time during which if something must be done, the UAV will have to do it since the human can’t.

We have to give the UAVs enough capability to make the right choice during The Gap. We have to

- Identify the possible decisions that could be, or must be, made.
- Identify what information is required to make those decisions.
- Ensure the proper information, and decision logic, is in place.

The Gap, Fig. 9, is important to us researchers since it, along with the task, drives vehicle autonomy. For instance, during postulated UAV aerial refueling The Gap is at least 7 seconds long—best case for satellite communications—the system has to be designed to handle itself during those seven seconds, especially if it is only feet away from a manned tanker. Increasing the UAV operator’s span-of-control only makes The Gap longer - as one increases the number of UAVs a human is operating it will take the operator more time to “get up to speed” on what any particular UAV is doing if help is required. Building on-board autonomy helps close The Gap since it off-loads some of the human decision making, allowing the human to focus on other problems for quicker resolution. Autonomy is Gap filler.



- **The Gap Is A Function Of Physics, Human Nature – Not Much Reduction**
- **For More UAVs Per Supervisor The Gap Gets Worse Even if System Autonomy Is Increased**
- **Effective Autonomy Increases Gap Which Can Be Tolerated, Or Increases Effectiveness With Same Gap**

Fig. 9 The gap.

5. Software—How Do We Develop And Test This Stuff?

What a UAV does, its courses of action, the decisions it makes, are determined by the software on-board which codifies knowledge. Therefore it is critical that this software be developed and tested to confirm its reliability (it does what we want it to) and safety (it doesn't do anything stupid). Easier said than done.

As we discussed in a previous paper,[11] the amount of flight critical code in UAV will rise drastically as we incorporate more pilot functions into it. This is shown below in Fig. 10.

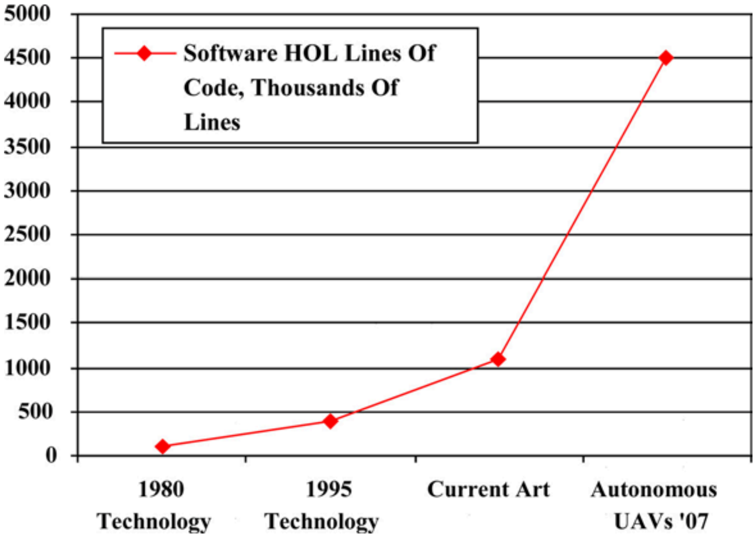


Fig. 10 Significant software growth due to autonomy.

So how do we test this stuff? At the time of this writing, all the author can say is that we do it in the traditional manner, leading to very significant code development costs. We hope to drastically reduce this in the future through research which is ongoing, both into how we deal with the code, and through use of technologies such as emergent behavior to reduce the amount of code. We need to do something. We cannot avoid the increase in code size as we increase autonomy, we must learn to deal with it.

6. Lesson Learned: Autonomy Blurs The Line Between Mission Management and Flight Control—How Do We Safely Partition The Code?

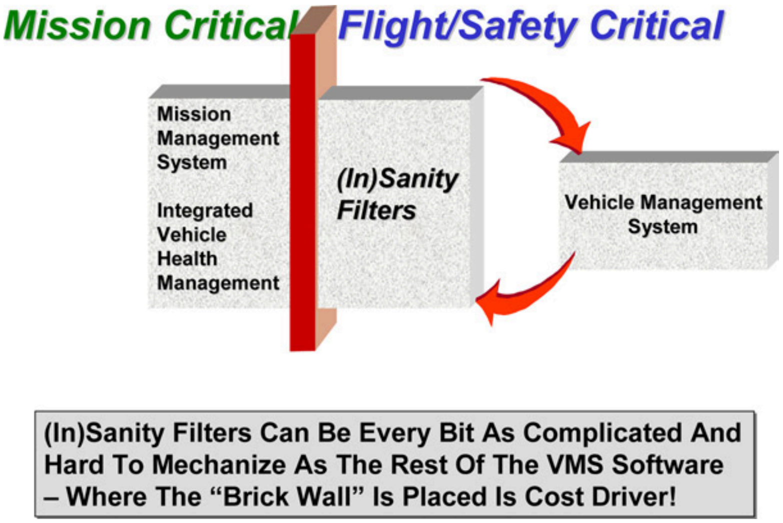
We want to take the human out of the aircraft, yet leave him in there, or at least the human's functions. One of those functions is to make decisions on what to do based on mission level information. This capability is now migrating to the on-board automation in UAVs Our vision of future vehicle management systems (VMS) is that they will manage both the flight and mission aspects of the UAV, therefore, they not only will process external mission information, but internally will contain both mission and flight/safety critical code.

Current best practice says that one does not need to test mission critical code as much as flight/safety critical code. One of the biggest drivers for this is the fact that before it is used to accomplish a flight/safety critical task (such as weapons release), it is put through a sanity check by a human.

But wait, in a manned aircraft mission systems are not as reliable as the VMS—the pilot acts as a sanity, or insanity filter, if you will, for the information he/she will be acting on. To recover this capability we need to add a filter as in Fig. 11, a filter for flight critical code that asks questions as the following:

- Is the trajectory plan free of collision hazards?
- Are the trajectory commands consistent with the mission state?
- Are the control commands within the vehicle's capability?
- Is the assessed health consistent with the observed actuator performance?
- Is the assessed health substantiated by current observations?

In autonomous UAVs one has to build that functionality back into the system, and that is very difficult since that can take as much (or more) logic, data, and internal models than that required to make the decision if the data is good in the first place. Since software is expensive one wants to minimize the filtering, but one can't make it all flight critical or cost becomes prohibitive also. Do we have the answer yet? No, but we're working at it.


Mission Critical
Flight/Safety Critical
Mission Management System
Integrated Vehicle Health Management
(In)Sanity Filters
Vehicle Management System

(In)Sanity Filters Can Be Every Bit As Complicated And Hard To Mechanize As The Rest Of The VMS Software – Where The "Brick Wall" Is Placed Is Cost Driver!

Fig. 11 (In) Sanity filters.

7. Lesson Learned: Software Development Technology Far Enough Ahead Of Software Certification Technology That Use Of Our Advanced Algorithms May Be Limited

Currently in our state-of-the-art research, we develop software applications that ride on top of middleware whose job it is to make the application software portable across different platforms. The middleware then interfaces the applications to the specific hardware and operating system. This is very efficient for most software applications; however, for flight/safety critical code it runs into a significant roadblock—certification procedure (guaranteeing real-time execution of flight critical functions). Figure 12 illustrates the difference between how we used to generate code, and how it's done today.

Usual avionics applications, especially flight critical code, were built into the operating system and hardware, "monolithic" code. Software testing procedures, the best industry practices, were designed based on this fact. Using those same methods on open systems applications is difficult at best.

What we would like to do is to separately certify the operating system and hardware, middleware, and applications, but we do not know how to devise requirements/tests to ensure that the software will meet safety and performance standards as it is moved across platforms, mixed with other "Plug & Play" applications and other flavors of middleware and hardware. Thus we have to treat, and test, the software as a monolithic block, erasing all of the affordability benefits of application-based code.

We do not believe this is an impossible situation to conquer. A year ago we initiated the VVIACS effort to investigate how we change our software development and testing paradigms, and we are making progress. Conferences in the near future will contain speakers discussing our progress towards a solution.

8. Lesson Learned: We Need To Become Comfortable With Non-Determinism

Intelligent autonomous systems are non-deterministic.[15] What we mean by this is that you might think you know what they will do, and in most instances you'll be right; however, you cannot rule out the fact that the system might do something totally different than predicted, and possibly hazardous. If it has autonomy, the decisions might not be what one expects.

We are currently researching the best way to put bounds on the non-determinism such that the outcome is never in question, but we give the system some latitude on how it handles a particular challenge. In other words, apply Asimov's laws to UAVs. But what causes the non-determinism?

Non-deterministic tendencies are either intentional or unintentional. Intentional non-determinism means that we have purposely included elements in the system that result in outputs that are not entirely predictable. Unintentional non-determinism comes from not knowing exactly how the system will be used, where it will be used, and the interactions between system components and with the environment.

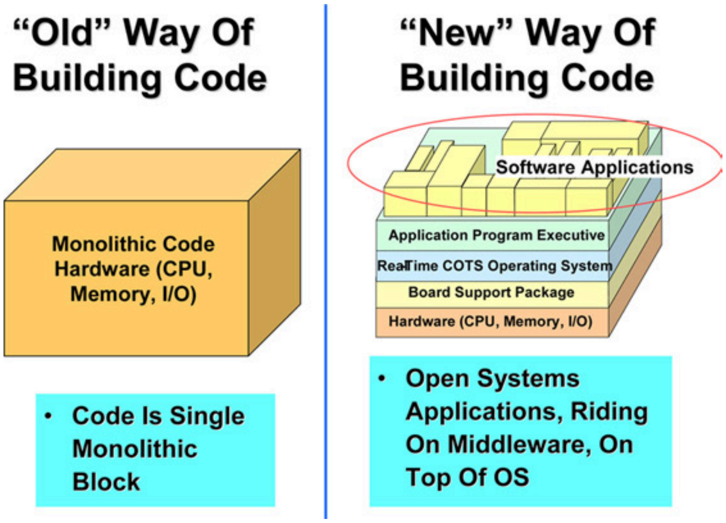

"Old" Way Of Building Code
Monolithic Code Hardware (CPU, Memory, I/O)
• Code Is Single Monolithic Block
"New" Way Of Building Code
Software Applications
Application Program Executive
Real-Time COTS Operating System
Board Support Package
Hardware (CPU, Memory, I/O)
• Open Systems Applications, Riding On Middleware, On Top Of OS

Fig. 12 Differences between how code used to be developed, and how it will be in the future.

Causes of intentional non-determinism (Fig. 13) include:

- Using negotiation strategies.
- Using emergent behavior.
- Allowing the system to learn.

Causes of un-intentional non-determinism (Fig. 14) include:

- Modeling errors.
- Uncertain environment.
- Latent faults in complex software.
- Information uncertainty.

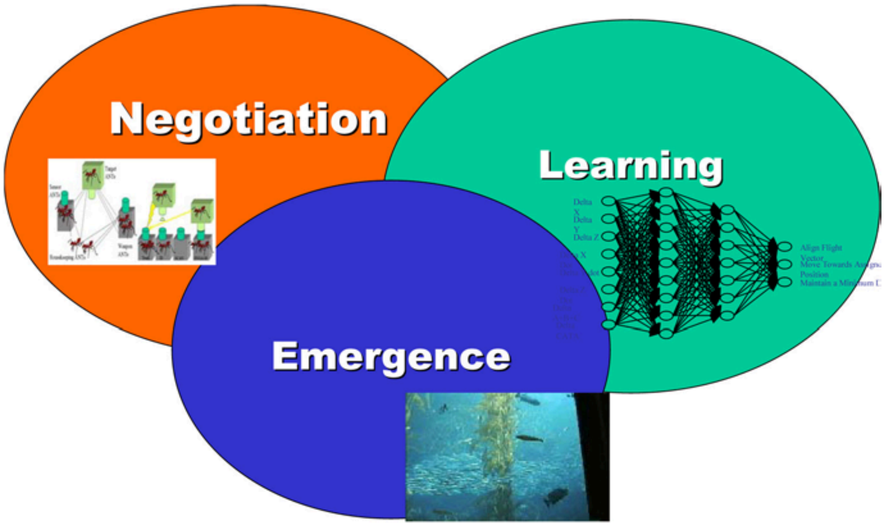
There is a very significant impact of our non-determinism. The existing paradigm is that if software has any non-deterministic aspects it cannot be certified. This is not explicitly stated in RTCA/DO-178B, the industry reference for airborne software certification.[16] The determinism aspect is probably a derived criterion from Section 4.5.c:

> *The software development standards should disallow the use of constructs or methods that produce outputs that cannot be verified or that are not compatible with safety-related requirements.*

The assumption is that anything that is not dead-on predictable cannot be verified. We need to get around current paradigms. We trust humans, arguably quite non-deterministic, with making decisions. We need to get away from perfection and define “good enough”. For our research into safety metrics the “good enough” level for bad decisions by the software is $1.8 * 10^{-5}$ failures per flight hour. The trick is how does one predict software failure rates and the failure consequences? Most regulators have deemed this impossible and throw out the idea of predicting software failure rates. We have taken the tack that we should look into the relationship between system and environment complexity, and the software failure rate and impact in the AFRL VVIACS effort. The reports aren’t back yet, but there is light at the end of the tunnel.

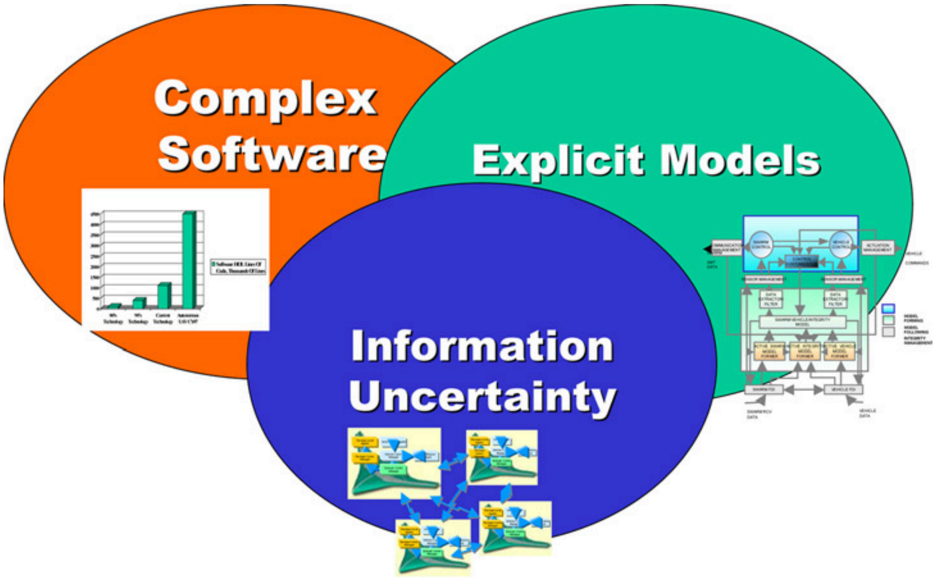
9. *Lesson Learned: Control Bandwidth A Function Of Human Trust Of UAV Autonomy*

The amount of bandwidth (BW) required to control the UAV, or UAV teams, is related to the amount of trust humans have in the UAVs. In general, the more trust one has in a UAV; the less information one needs to “second-guess” the UAV. However, the bandwidth one might need at any particular moment (latent bandwidth) still can be considerable, and is linked to the task the UAV is doing.



…results from the intentional inclusion of chaotic stochastic elements and elements relying on future events

Fig. 13 Causes of intentional non-determinism.



…results from the interplay of complex software, explicit models, and uncertain information

Fig. 14 Causes of unintentional non-determinism.

Figure 15 shows three cases, with increasing UAV autonomy as one goes from left to right.

- To the left, systems with no autonomy need to get relatively high BW commands from the control station, they send back as much data as practical to give the operator full situational awareness.
- As we give the system a bit of autonomy, our command BW from the operator drops off since fewer commands have to be issued, and less situational data is required back from the UAV except for that required to support those decisions the operator reserves. For that, large bandwidths may be needed, so there exists a "shadow"

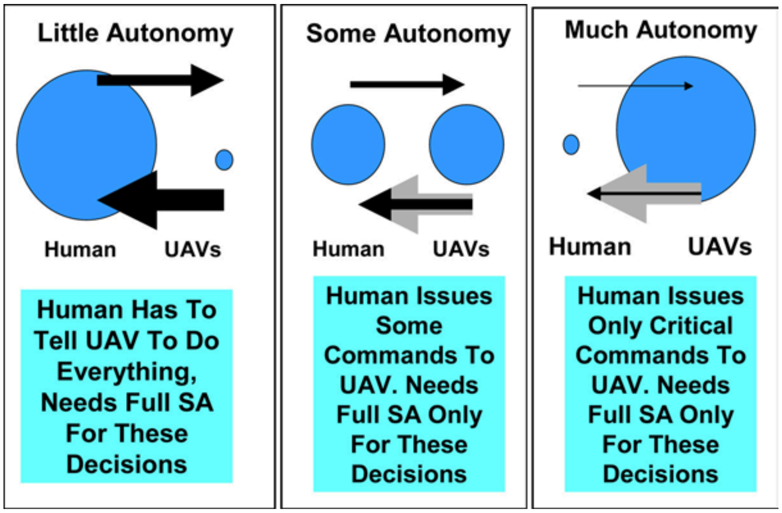
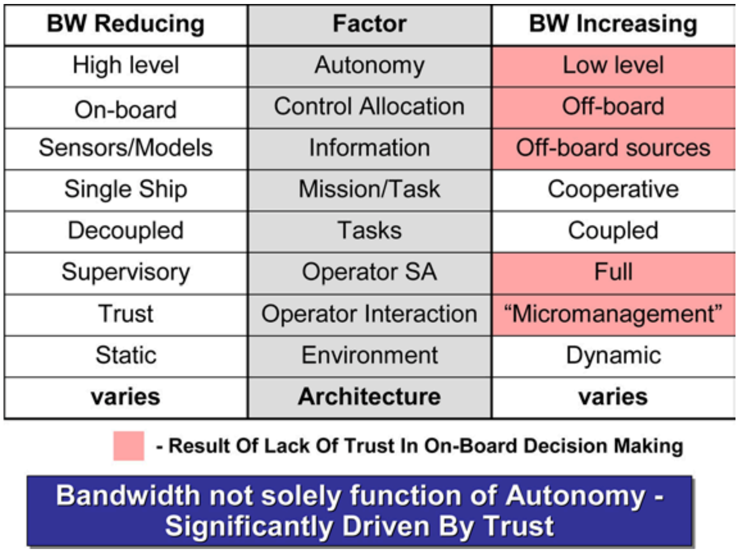

Little Autonomy
Human
UAVs
Human Has To Tell UAV To Do Everything, Needs Full SA For These Decisions
Some Autonomy
Human
UAVs
Human Issues Some Commands To UAV. Needs Full SA Only For These Decisions
Much Autonomy
Human
UAVs
Human Issues Only Critical Commands To UAV. Needs Full SA Only For These Decisions

Fig. 15 Impact of autonomy, and human trust in it, to required comm bandwidth.

Table 1 BW tied to trust.



BW Reducing	Factor	BW Increasing
High level	Autonomy	Low level
On-board	Control Allocation	Off-board
Sensors/Models	Information	Off-board sources
Single Ship	Mission/Task	Cooperative
Decoupled	Tasks	Coupled
Supervisory	Operator SA	Full
Trust	Operator Interaction	"Micromanagement"
Static	Environment	Dynamic
varies	**Architecture**	**varies**

🟥 **- Result Of Lack Of Trust In On-Board Decision Making**

Bandwidth not solely function of Autonomy - Significantly Driven By Trust

or latent BW requirement for the human to make those decisions the UAV is not equipped to make. This bandwidth isn't needed all the time, but we must make allowances for it.

- The trend continues as we increase the autonomy. The required BW shrinks except for those decisions kept by the human, and for those a latent BW requirement is there equivalent to the amount of data and rate the human operator needs to make the decision.

This latent communication requirement, even for significantly autonomous systems, could be quite large if hyperspectral data is needed to make the decision. It exists as long as the human doesn't trust the UAV to make the decision—BW is a function of trust [*Note 4*].

Table 1 is another way of looking at how bandwidth is driven by trust of the UAV operations. It shows factors in UAV control which influence control bandwidth. What the lack of trust impacts is highlighted in pink. We don't give it autonomy, we make decisions off-board, we rely on off board data (don't trust the sensors), we need to know everything that is going on, and we want to be involved in every decision made. Again, for control, trust drives bandwidth.

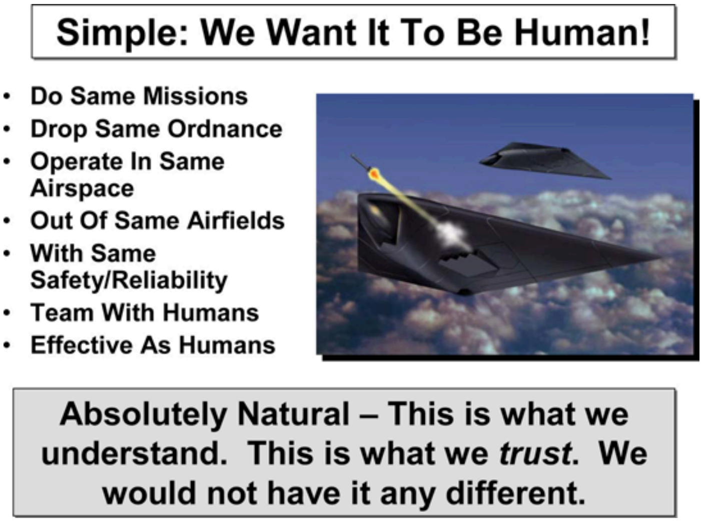
C. Make Them Affordable

1. *Overall UAV Affordability-You Will Pay For What You Want, But Autonomy Helps*

To put it bluntly, UAV unit costs are rising sharply. Although there are many groups tasked to look into this, precious few have discovered the real reason for the price rise. We want the UAVs to be human, something obvious to autonomy developers and shown in Fig. 16. Going through the latest DOD UAV Roadmap[5] one can note items removed and added to the UAV affecting the cost. Some of these are shown in Fig. 17. Although we are taking the human out of the cockpit, we want the human capability to remain in the UAV. When we give the UAVs human safety, human capability, human airspace operations equivalence, we will have given them human aircraft costs.[3]

Since aircraft are not bought based on total ownership costs for the capability they bring over their lifespan, but on unit costs at the time of construction, having UAVs cost the same as manned aircraft can be damaging to the UAV's chances of funding. We need to accept the equivalent unit price for equivalent capability and work to reduce total ownership costs.

Our contribution to lower unit costs is greater autonomy—the ability for fewer humans to operate more UAVs. Humans are the most expensive part of any weapon system, and any goal of increased affordability should evaluate how we use fewer humans to achieve the same effect. Our goal is to turn human UAV operators into UAV supervisors,



Simple: We Want It To Be Human!

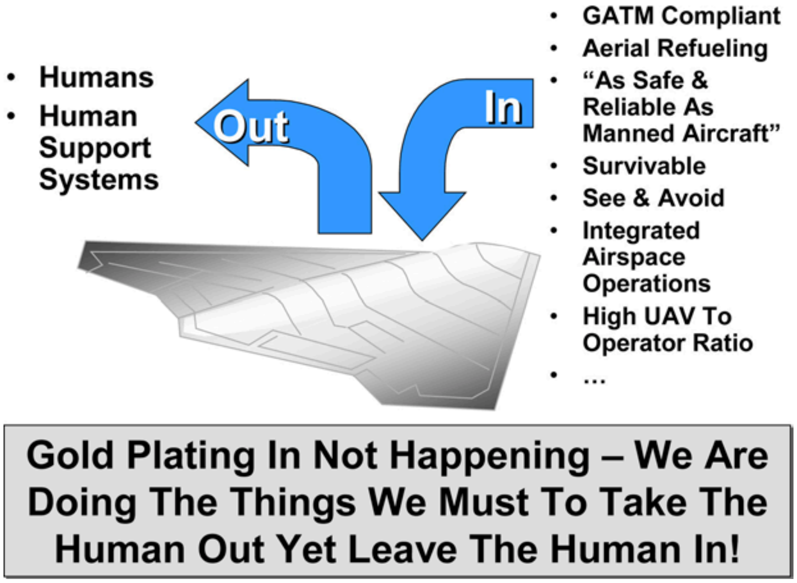
- Do Same Missions
- Drop Same Ordnance
- Operate In Same Airspace
- Out Of Same Airfields
- With Same Safety/Reliability
- Team With Humans
- Effective As Humans

Absolutely Natural – This is what we understand. This is what we *trust*. We would not have it any different.

Fig. 16 Why UAVs are getting expensive? Why isn’t the UAV cheap anymore?

increasing both the number of UAVs, or UAV teams, a human can direct, and increase the level at which the human directs, from tactical to strategic levels, and let the UAVs worry about the decisions below that level. In other words, increase UAV autonomy.

Our autonomy development efforts are aimed at reducing the need for humans to be involved in the warp and woof of UAV operation. Are there limits? Possibly. If one looks at AWACS controllers directing manned aircraft, or air traffic controllers doing similar directing one can get a very rough idea on how many autonomous vehicles a human can handle. We don’t know the hard number, but we are looking.



- Humans
- Human Support Systems

- GATM Compliant
- Aerial Refueling
- “As Safe & Reliable As Manned Aircraft”
- Survivable
- See & Avoid
- Integrated Airspace Operations
- High UAV To Operator Ratio
- …

Gold Plating In Not Happening – We Are Doing The Things We Must To Take The Human Out Yet Leave The Human In!

Fig. 17 Gilding not happening, replicating capability is.

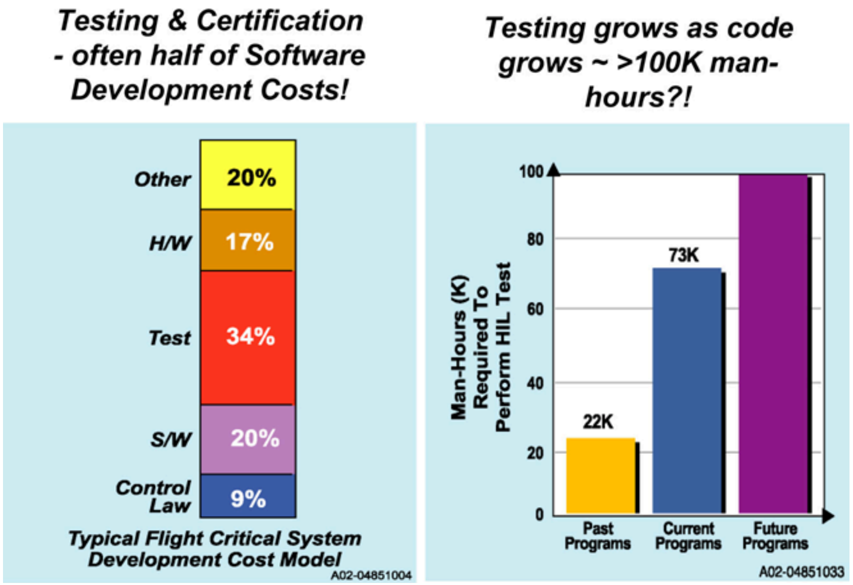


Fig. 18 Predicted growth in code cost due to testing autonomous functionality.

2. Lesson Learned: Software Significant Cost Driver For Current Systems—Only Gets Worse For Autonomous UAVs! Software Safety and Affordability Are Linked

As mentioned in the safety segment of this paper, software complexity is rising for all systems as functionality is increased, and this is impacting how we test software. What wasn't related in that section was the attendant cost of this increased software functionality—our increasing the autonomy of the UAVs.

Figure 18 contains two charts. The first shows that over a third of the current cost of flight control program is testing. As software becomes more complex, this percentage will increase.

We have already shown in Fig. 10 projected increases well beyond what we currently have in VMS software size. What wasn't shown, and is illustrated in the right half of Fig. 18, is the increased testing time for this increased complexity software in terms of man-hours required to prove the code performs safely. However, this isn't the total cost impact to the weapon system from the software. Probably the highest cost of the software is the wait that the rest of the system must endure while software is being tested, especially for unintended consequences (remember the non-deterministic discussion, this really rears its ugly head dealing with software modifications and retests). Software, especially flight and mission critical software, can stop a program dead in its tracks, with overhead charges piling up while the software is fixed.[17,18] As our software becomes more complex, as we increase a UAV's autonomy, the risk for significant program delays becomes increasingly greater.

As with the challenge to safety due to software, we are meeting this one head on. Since the investment one spends on software testing is related to how safe one can prove it is, one cannot divorce software safety from affordability. We have recognized this fact and are devising ways to make the software testing required to prove safety affordable. Programs are underway, but it is too early to issue our report card.

Software is critical to UAV autonomy, and also is becoming its major cost driver. Not only is it the driver for system testing costs, its complexity is getting to the point where we seriously doubt our ability to adequately test it. Changes in development and testing methods and philosophies are required.

IV. Where Are We Heading From Here?

So what is next? The next series of bulleted comments reflect on where we want to head to in the next few years, outlining questions we want to answer, and research directions we are heading in.

- We know quite a bit about building leader-follower UAV teams. Now we need to do more "what if's" using that knowledge. What happens when we bring in the warfighter to discover new ways of employing autonomous UAVs? What paradigms must we jettison?
- We are looking at flattening the UAV Team "management structure" by developing distributed control technology but we don't know how far we can stretch that given practical communication constraints. There is a reason humans form leader-follower hierarchical structures. What is it and won't our UAVs have to follow it? Is distributed control actually better for some tasks, or do the communication and other required items to do the task swamp our savings?
- We need to build human trust in UAV operation. Without trust, the human won't let the UAV do anything by itself, so the notion of building more autonomy doesn't make any sense without working the trust issue. This cannot be worked from the UAV side alone; we must team with human effectiveness research and development folks to ensure a trust "handshake" between UAV and human. We need to build the conversations.
- We need to get away from our paradigms on how to build and test software as well as how software should act. Non-determinism is a facet of every intelligent system and will be so for our UAVs. We need to bind the non-determinism in an "Asimovian" sense.
- Finally, we need to increase UAV autonomy to increase their affordability. Moving human/UAV interactions from the specific task, to the tactical, and then to the strategic level, widening the human's span-of-control will reduce the number of humans required to control UAVs, transforming operators into supervisors.

How well will this work? How well will it play out? Will we make our goals? Ask again in three years.

Acknowledgments

The author would like to thank Mr. Robert R. Smith Jr. for his support in writing and reviewing this paper, and my wife Alice for putting up with the typing on the nights and weekends.

References

[1] Clough, B., *Unmanned Aerial Vehicles: Autonomous Control Challenges, A Researcher's Perspective*. World Aviation Congress 2000, San Diego, CA. October 2000.

[2] Clough, B., *Metrics, Schmetrics! How the heck can one figure out how autonomous a UAV is?* AIAA 1st Unmanned Systems, Technologies, and Operations Conference and Workshop. Portsmouth, VA. May 2002.

[3] Clough, B., "*UAVs—You Want Affordability And Capability? Get Autonomy!"*. Proceedings, AUVSI Unmanned Systems 2003. AUVSI Jul 2003.

[4] Col John Boyd's OODA Loop can be found at www.d-n-i.net/second_level/boyd_military.htm

[5] *UAV Roadmap 2002*. Department of Defense. 2003.

[6] Clough, B., *Relating Autonomy to a Task—Can it be done?* AIAA 1st Unmanned Systems, Technologies, and Operations Conference and Workshop. Portsmouth, VA. May 2002.

[7] Clough, B., *UAV Swarming? So What Are Those Swarms, What Are The Implications, And How Do We Handle Them?* AUVSI Unmanned Systems 2002. Symposium, Lake Buena Vista, FL,. July 2002.

[8] Loscher, J., *Application of Neural Networks in Flight Control Technology*. AIAA Missile Sciences Conference. Monterey, CA. November 2000.

[9] McDowell, J., Smith, R. *Agent-Based Hierarchical Architecture For Autonomous Control of Lethal Unmanned Vehicles*. Horizons Magazine, Air Force Research Laboratory, Wright-Patterson AFB, OH 2002.

[10] From an email conversation with Mr. Robert Smith Jr., co-worker at the Air Vehicles Directorate.

[11] Clough, B., *Autonomous UAV Control System Safety—What Should It Be, How Do We Reach It, And What Should We Call It?* NAECON 2000 paper. October 2000.

[12] O'Neil, W. and Chen, W. *See and Avoid Sensor System Design Parts I & II*. Dual papers written for the 2nd AIAA Unmanned Systems, Technologies, and Operations Conference. AIAA. San Diego, CA. September 2003.

[13] From MIL-STD-9490-D.

[14] Asimov, I., *Runaround*. Astounding Science Fiction magazine. 1942.

[15] Clough, B., *Autonomous UAVs—Give Them Free Will And They Will Execute It!* Proceedings of the AUVSI Unmanned Systems 2001 Symposium. Baltimore, MD. 2001.

[16] RTCA/DO-178B "*Software Considerations in Airborne Systems and Equipment Certification"* Radio Technical Commission for Aeronautics. Washington, DC. 1992.

[17] "*F/A-22 avionic fixes make progress in lab tests, not yet in flight*". Aerospace Daily, Aviation Week. 19 Feb 2003.

[18] "*Software Failures Delay F-22 Tests, U.S. Tester Says*" Tony Capaccio, Bloomberg.com. 30 Jul 2002.

Notes

1. One of the comments by a warfighter was that "I want the UAV to do everything I tell it unless I tell it something stupid".
2. This was done using tools from Integrated Product Process Development that is being embraced by the Air Vehicles Directorate. More and more of our 6.2 (engineering development) work will use parts of IPPD to ease transition to operational systems, having demonstrated affordability improvements before it leaves the lab.
3. The DOD DDR&E Fixed Wing Vehicle Initiative (FWV) has top-level vehicle reliability goals, we originally derived our LOC_{ac} rate as an internal attempt to see what we would have to prove to get to the FWV metric. Other players in the FWV saw our results and embraced them also.
4. Bandwidth is a function of trust is not just for UAVs with some level of autonomy, the same holds true for humans. One's time spent with a subordinate that isn't trusted is quite large compared to those who are. This relationship is the same for either carbon or silicon based units.