A Critique of the "Unmanned Systems Safety Guide for DoD Acquisition"

R. D. Alexander, PhD, T. P. Kelly, PhD; University of York; York, UK

N. J. Herbert, BSc; BAE Systems Military Air Solutions; Preston, UK

Keywords: unmanned systems, autonomy, guidance

Abstract

The Unmanned Systems Safety Guide for DoD Acquisition (henceforth, "the guide") is intended to help safety engineers achieve safety in unmanned systems acquisition projects. Given the rising number of such projects, this is a worthy goal. The guide contains some sound advice, but it also has a number of weaknesses. In particular the core elements of the guide (the Top Level Mishaps and the Precepts) are poorly defined, and the choice of what to include is odd. Using the guide in support of a risk-based safety process is difficult because of the lack of explicit rationale. The way that the guide is structured may make it cumbersome to maintain in future. We identify a number of ways in which the guide could be improved, and highlight some key opportunities for future development.

Introduction

The Unmanned Systems Safety Guide for DoD Acquisition (ref. 1) was issued in July 2007. It aims, in its own words, *"to ensure the design and development of UMSs* [Unmanned Systems] *that incorporate the necessary safety design rigor to prevent potential mishaps, or mitigate potential mishap risk"* and to *"consider real and potential Concepts of Operation (CONOPS) of UMSs and establish fundamental operational safety requirements necessary to support safe operation of the UMS.*" In other words, the guide is meant to help safety engineers and UMS operators design and operate them in a way that avoids accidents.

The authors are firmly behind the intent of the guide, and believe that now is a critical time for such guidance to be written. However, there are a number of problems with the guide that concern us. In this paper, we will explain the issues we have identified, and highlight some strategies for resolving them. Some of this, we hope, goes beyond the UMS guide in particular and into broader questions of what safety guidance should contain.

In the next section, we provide a brief overview of the guide and its contents. The following two sections discuss the two most problematic parts the guide, explaining the grounds for our concern. We then address some potential future problems stemming from the structure of the guide (particularly, from its choice of what to include and exclude). Finally, we make some suggestions for how future revisions could resolve the issues we've identified.

A Brief Summary of the Guide

The first part of the guide is a set of overviews, discussing a range of UMS-relevant issues a free text format. The discussion makes up only a small part of the page count; the bulk is in the appendices, particularly in the definitions of terms and in the tables describing the safety precepts. The general discussion is broken down into several topics. First, key terms and principles are defined. This is followed by sections on system safety in general and on UMS safety specifically. The latter then expands into sections on the safety considerations at the program, design and operational levels. The discussion is a good overview of the area and highlights a number of important points.

A key element of the discussion, for our immediate purposes, is a set of nine Top-Level Mishaps (TLMs) that are introduced as things that we want to avoid. To this end, a set of *safety precepts* are identified. As will become apparent, it is with the TLMs and the Precepts that we have the most concerns.

The apparent aim of the guide is to condense expert knowledge. There are more than forty authors credited, drawn from a wide range of backgrounds including all US armed services, the Department of Defense, major defense contractors, and academia. It's well known that domain knowledge is crucial for effective engineering, and with

UMS as a fast-growing area useful domain knowledge is thin on the ground. The guide has the potential to distribute the limited available knowledge (and hard-won experience) to a wide audience.

The guide's aim is important, and its general approach (condense expert knowledge) is credible. Given the rapid expansion in the use of UMS, now is the time when we really need such guidance.

Top Level Mishaps

A TLM is defined to be:

> *"…a generic mishap category for collecting and correlating related hazards that share the same general type of mishap outcome event. A TLM is a common mishap outcome that can be caused by one or more hazards; its purpose is to serve as a collection point for all of the potential hazards that can result in the same overall TLM outcome, but have different causal factors…"* (ref. 1)

The authors have great difficulty in understanding the above definition. It is certainly possible that this represents a conflict of US vs. UK terminology – perhaps the above is clear and concise to a typical US safety engineer. However, we have been unable to come to a satisfactory understanding. This confusion can be made more explicit by providing a categorisation of the nine identified TLMs in terms that are familiar to us. This is shown in Table 1.

Table 1 - Top Level Mishaps

TLM-1	Unintended/Abnormal system mobility operation	HAZARD
TLM-2	Inadvertent firing or release of weapons	HAZARD
TLM-3	Engagement/Firing upon unintended targets	HAZARD
TLM-4	Self-damage of own system from weapon fire/release	ACCIDENT
TLM-5	Personnel injury	CONSEQUENCE
TLM-6	Equipment damage	CONSEQUENCE
TLM-7	Environmental damage	CONSEQUENCE
TLM-8	Vehicle loss	CONSEQUENCE
TLM-9	Vehicle collision	ACCIDENT

It can be seen that the TLMs are a collection of hazards, accidents and generic consequences of accidents. Certainly, all of these things are important, and are to be avoided. Getting confidence that we *have* avoided them, however, requires confidence that our set is complete, and this is very difficult to do when the examples of set members are overlapping. Indeed, it's easy to place the different subtypes of TLMs at different levels of abstraction: the four "consequences" are the top-level things that we want to avoid, the two accidents are events that could directly cause those consequences, and the three hazards are things that could cause those accidents. It is not clear how we might argue that we have identified all TLMs when the set of things that count as TLMs are so varied.

We don't mean to suggest that the TLM definition should achieve rigorous mathematical precision. But it should be sufficiently clear that an experienced safety engineer can determine if something belongs in the set or not.

The Precepts

The guide states that *"the various safety precepts developed for UMS have been specifically directed at resolving one or more of the TLMs"*. It does not, however, present any explicit mapping from the TLMs to the individual precepts. This is unfortunate, because it is not clear how the application of any particular precept supports a claim that the UMS is free from a particular TLM.

Viewed in the context of a UMS-specific safety guide, some of the individual precepts are poor. Probably the worst example is DSP-11 *"The UMS shall be designed to minimize the use of hazardous and toxic materials"*. As a general precept for equipment safety, this is sound and necessary. In a safety standard ostensibly dealing with the safety of a novel, high-risk class of systems, it has no place at all. It actually serves as a distraction from the UMS-specific safety guidance. The entire precept is completely generic; there is nothing in its body text that is specific to UMS.

DSP-9 stands out as being both generic and overly narrow: *"Safety critical software for the UMS design shall only include required and intended functionality"*. The precept's body text expands this by referring to "dead code", defined there as *"code never intended for any system use"* (note how this differs from common usage – "dead code" is normally means code that can never be executed). This is the only one of the 19 design precepts that deals with general software quality issues, yet it limits itself to one narrow aspect. (The *programmatic* precept PSP-3 also makes some reference to software quality, e.g. through testing.) Nothing in the precept justifies why unused functionality would be a particular concern for an autonomous system. We could, perhaps, create such a justification (one possibility is that the additional functionality could be available for autonomous use by the system, e.g. it could add additional actions for a planner to consider) but this is not particularly compelling. It is not clear why this issue was given a whole precept when other (more challenging) software quality concerns were not.

The guide motivates DSP-9 with the example of the Therac 25 radiotherapy machine. Comparing the paragraph in the guide with the thorough review of the Therac 25 accidents by Leveson (ref. 2), the relevance of the precept to the accident is questionable. Although Leveson does note that poorly managed code reuse was a contributory factor in the accidents, the most severe accident (the 'Yakima' accident) was caused by a concurrency problem, not by "dead code" of any kind. Concurrency errors are much more difficult to detect, and require much more sophisticated testing and analysis methods, than the mere presence of unwanted code. The use of this example for this precept is worrying in that it distracts from the real challenges that the Therac 25 accidents reveal.

Some of those precepts that are more relevant are quite basic. For example, DSP-13 is *"The UMS shall be designed to identify to the authorized entity(ies) the weapon being used or fired, but prior to weapons release or fire."* Operators may benefit from such information, and it may help them to avoid some errors, but this is only a small part of what they need to use weapons safely. By contrast, the single precept DSP-3 (*"The UMS shall be designed to provide information, intelligence, and method of control (I2C) to support safe operations"*) attempts to cover the operator's entire (safety) requirement for situational awareness. It is unclear why DSP-13 is seen as requiring an entire precept, while DSP-3 covers so much.

Those precepts that are highly relevant are underdeveloped. For example, OSP-1 states that *"The controlling entity(ies) of the UMS should have adequate mission information to support safe operations"*. The body of the precept gives examples of information that the operator may need (e.g. mission objectives, CONOPS, weather conditions). Here, however, it is the detail that matters – the definition of 'adequate' will depend heavily on the system and the situation. Stating that 'adequate' information is needed is of little value – instead, developers need hard criteria for adequacy and a way to derive the information requirements in a given instance.

One area where this lack is very apparent is *autonomy*. The guide mentions potential for autonomy as part of its definition of a UMS (*"An electro-mechanical system that is able to exert its power to perform designed missions and includes the following: … (2) manned systems that can be fully or partially operated in an autonomous mode"*),

and a range of autonomy issues are discussed in the general discussion text. The precepts, however, offer little to address these issues.

Most of the guide is not UMS-specific. The majority of the 30 precepts are generic, being relevant to many non-UMS situations, and are marked as such in the guide. A naïve reader might infer from this that engineering safe UMS is little different from engineering safe manned systems. This is a dangerous inference – there is a serious risk that the guide's generic nature will distract readers from the genuine difficulties and challenges of UMS safety.

These criticisms aside, the guide provides a range of important details. Most of these are contained in the "Examples" and "Detailed Considerations" sections of each precept. The details represent the combined experience and foresight of the many authors credited in the guide. For example, DSP-7 includes *"Transition from a training mode to an operational mode necessitates clearing registers, otherwise invalid tracks could be left in the weapons control system"* as an example and DSP-8 includes *"For range testing or training consider the need for multiple operators to have ability to cease fire or abort weapons fire"* as detailed consideration. These valuable insights are, however, easy lost in the sea of the vague and generic.

A cost of generality is lack of direction. The guide is, we presume, aimed at experienced safety engineers and system operators. These are not people who need to be educated about safety in general (if they *do* need this, then maybe we need new engineers and operators who are worthy of their titles). Rather, they need to know what's new about UMS, and how they need to change their thinking and behavior when they move onto a UMS project.

For an experienced safety engineer, there is little new in the precepts. There is little that tells them about new challenges, little telling them how to re-prioritize their standard safety concerns. One could question what they will get to repay their investment of time and effort in studying the guide.

The Format of the Guide

There are two problems with the format and structure of the guide. These are the lack of explicit rationale and the high proportion of generic content.

Lack of Rationale: There is little explanation given as to how the precepts fit together to collectively prevent the TLMs. Each precept has a "Rationale" entry, but most of these are very basic. For example, the rationale for DSP-4 (*"The UMS shall be designed to isolate power until as late in the operational sequence as practical from items such as: a) Weapons, b) Rocket motor initiation circuits, c) Bomb release racks, or, d) Propulsion systems."*) is *"The intent of this precept is to preclude the inadvertent release of hazardous energy."*. Few of the "Rationale" sections make links between precepts.

It is well known that maintaining knowledge of rationale is critical for engineering over the long term. See, for example, the discussion by Leveson in (ref. 3). If an engineer knows the reason for a requirements or a design decision, they can reason about whether it can be changed. If there's pressure to change it, they can use the rationale to work out how important it is, allowing them to trade off against competing demands. If they don't know the reason for a decision, they can't do this kind of reasoning. This leaves them with the choice of assuming it can be safely changed (and risking adverse consequences when it turns out it couldn't) or assuming it can't be changed (and risking getting stuck with something they don't need). Rationale is also important for outsiders (like the authors of this paper) who seek to understand decisions and provide critique.

It is sometimes possible to reconstruct rationale. Indeed, this is often necessary when a standard or system design comes to be revised. An example of an extensive reconstruction effort is that by Hall-May in (ref. 4). There, Hall-May takes the UK Rules of the Air (general regulations for civil aircraft) and tries to discover the reasons behind them. He achieves this by expressing the rules as a goal structure, working downwards from high-level safety goals (e.g. *"Avoid aerial collisions"*) to specific rules that cause aircraft to meet those safety goals (e.g. *"Aircraft are forbidden to turn when overtaking"*).

Following on from Hall-May's example, the authors have created an equivalent goal structure for the UMS safety precepts. This was fairly difficult, as we had to infer the intent of each precept from context, and from our own model of system safety. The structure we ended up with seems to show substantial gaps, in that several safety issues are not addressed at all. It is difficult to be sure, however, because we don't know the intent of the precept creators. There may be an overarching concept that forms the precepts into a coherent whole. Alternatively, there may be a conscious scoping decision that makes a clear boundary between the contribution of the precepts and the work that needs to be done by individual project teams. Either way, we can't find this in the guide.

High Proportion of Generic Content: As noted above, many of the precepts are not UMS-specific, or are only in parts. This increases the likely workload for future maintenance of the guide – any change to overarching standards (e.g. MIL-STD-882) or general safety guidance will necessitate change in the UMS guide. Were the guide to restrict itself to UMS-specific guidance, only major structural changes in the general documents would impact it.

A second cost of generic content is in reader goodwill. A busy safety engineer or system operator will appreciate guidance on how to deal with new and difficult situations. He may not appreciate being told things he already knows. A slim, focused pamphlet giving "What You Need to Know About Unmanned Systems" is more valuable than another weighty standard.

This issue goes beyond the current case of the UMS guide to a general principle: if you can modularize guidance, it can be reviewed, maintained and understood more easily. This is well-known for software, and it should apply to complex technical documents as well.

We would hold up the UK defense equipment safety standard, Def Stan 00-56 (ref. 5), as a good example of modularity. It has a concise mandatory component ("Part 1"), which must be followed by all projects, and a larger guidance component ("Part 2"). The guidance gives developers suggestions on how to comply with the mandatory part, and remains generic in its applicability (with some expanded sections for programmable electronic elements). Further guidance is now being developed for certifying software under 00-56 – this specialized guidance will be supplementary to the generic guidance. Eventually, it is likely that a guidance document will be produced for unmanned and autonomous systems. By couching this in terms of the core standard, the core guidance, and the software-specific guidance, it will be able to focus clearly on just those issues that matter for UMS.

Ways Forward

In the light of the critique above, we can identify some fruitful avenues for development in future revisions of the guide.

Trim Down to the Essentials: The guidance should be pared back to focus on the aspects of UMS that are new, surprising, or particularly challenging. It should be placed in a context of existing, generic, standards and guidance. In essence, it should state "For safety, the difference that UMS make is…"

This change would be valuable in three ways. First, it will help with learning and teaching, because there will be less for readers to take in and what's there will be more obviously new. It will be useful for criticism, because the essence of the guide will be more obvious (as distinct from what it has inherited from other DoD standards). Finally, it will reduce the maintenance load because only major changes in general safety thinking will require changes to the guide.

If the current guide was stripped down to the UMS-specific core, it might seem rather thin. However, in time, and with growth in UMS knowledge, the "weight shortfall" will easily be filled.

Address the Autonomy Issue: Although the general text of the guide addresses a number of aspects of autonomy, this is almost completely absent from the precepts. If the precepts cannot easily be updated to address this, then they would benefit from a qualifier which restricts their applicability. An example of this can be found in the UK military UAV standard Def Stan 00-970 Part 9 (ref. 6), which explicitly restricts its own applicability to UAVs that operate using a pre-planned flight path which is uploaded to the UAV and which can be changed (by the operator) at any

time during flight. It also stipulates that for UAVs under its remit *"direct, online control of the UAV flight path shall be avoided where possible"*.

Eventually, it might be sensible to split the guide into a core that covers all UMS and a supplement for "highly autonomous systems".

Explicate Rationale: As noted above, the lack of clear rationale is problematic for several reasons. By making the rationale for the precepts explicit, the guide can be opened up to easier review, criticism and maintenance.

Presenting the rationale as a goal structure will allow easy criticism and assessment of completeness. Developers in the UK would be particularly interested in safety case fragments or safety case patterns, but this is obviously outside the intent of the guide.

Rework TLMs to be of a Single Type: The current mixture of entity types in the TLM list makes it difficult to assess and argue about completeness. It would be useful to rework them so that they can be clearly seen to be members of one distinct category.

The revised TLMs should be at a level that is somewhat UMS-specific. As we have said elsewhere, non-specific advice belongs in generic safety guides. Using our categories from Table 1**,** we suggest that "hazard" would be the most appropriate level for the entries in a revised set of TLMs. The "consequences" are completely generic (not UMS-specific in the slightest), and the accidents are likely to be much the same as for the equivalent manned systems. It is the level below, the level where typical UMS mechanisms are visible, where we will see a distinction (and hence need to work differently).

An alternative to redefining the TLMs would be to create an intermediate level towards which the precepts could be targeted. Section 3.1 of the guide presents a list of "safety concerns" for UMS, immediately prior to the presentation of the TLMs in Section 3.2. As an intermediate layer between accidents and precepts, this list is a reasonable candidate. It is still of mixed type, but many of its members are UMS-specific (e.g. *"Loss of UMS ownership (lost out of range or to the enemy)"*), or are of particular concern for UMS (e.g. *"UMS in indeterminate or erroneous state"*). It is not ideal, but it is useful; the current TLMs are neither.

Conclusion

In summary, the guide is valuable contribution at an important time. It provides many insights about UMS in a form that is accessible (particularly for US DoD personnel and contractors, who are its primary audience). It is of high relevance to safety engineers outside the US, particular in those jurisdictions where the use of safety cases makes it easy to adapt and adopt such "foreign" guidance.

The guide's strengths, however, risk getting lost in its weaknesses. The good advice it contains is overshadowed by a large amount of generic safety content. In particular, there is a risk that the insightful, UMS-specific discussion in the main body of the guide will be ignored in favour of the very generic content of the precepts. A working safety engineer might come to the guide looking for help with emerging UMS problems - we are concerned that he would be annoyed by the paucity of new content.

We have identified some ways by which these issues can be rectified, and there are good prospects for future revisions.

References

1. Unmanned Systems Safety Guide for DoD Acquisition, US Department of Defense, 2007.
2. N. G. Leveson, Safeware: System Safety and Computers: Addison-Wesley, 1995.
3. N. G. Leveson, Intent Specifications: An Approach to Building Human-Centered Specifications, IEEE Trans. on Software Engineering, 2000.

4. M. Hall-May, Ensuring Safety of Systems of Systems—A Policy-based Approach, PhD Thesis, University of York, 2007.

5. MoD Interim Defence Standard 00-56 Issue 4 - Safety Management Requirements for Defence Systems, Ministry of Defence, 2007.

6. Defence Standard 00-970 Design and Airworthiness Requirements for Service Aircraft Issue 4 Part 9 — UAV Systems, Ministry of Defence, 2006.

Acknowledgments

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

Biography

Dr Robert Alexander, Ph.D., Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK, telephone – +44 1904 432792, facsimile – +44 1904 432767, e-mail – robert.alexander@cs.york.ac.uk

Dr Rob Alexander is a Research Associate in the High Integrity Systems Engineering (HISE) group in the Department of Computer Science at the University of York. His research focus is on safety engineering for unmanned and autonomous systems, although he is active in a range of other areas including System of Systems engineering, software safety, and real-time risk awareness. He is also involved in research on the evaluation and accreditation of complex simulation models. Rob has published papers in international conferences on a range of systems safety issues. He graduated from Keele University in 2001 with first class honours in Computer Science, and was awarded his doctorate in 2008 by the University of York.

Dr Tim Kelly, PhD., Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK, telephone – +44 1904 432764, facsimile – +44 1904 432708, e-mail – tim.kelly@cs.york.ac.uk

Dr Tim Kelly is a Senior Lecturer in software and safety engineering within the Department of Computer Science at the University of York. He is also Academic Theme Leader of the UK MoD Software Systems Engineering Initiative Dependability Theme. His expertise lies predominantly in the areas of safety case development and management. His doctoral research focused upon safety argument presentation, maintenance, and reuse using the Goal Structuring Notation (GSN). Tim has provided extensive consultative and facilitative support in the production of acceptable safety cases for companies from the medical, aerospace, railways and power generation sectors. Before commencing his work in the field of safety engineering, Tim graduated with first class honours in Computer Science from the University of Cambridge. He has published a number of papers on safety case development in international journals and conferences and has been an invited panel speaker on software safety issues.

Ms Nicola Herbert, BSc., Senior Systems and Software Engineer, BAE Systems, Military Air Solutions, Warton Aerodrome, Preston, Lancashire, PR4 1AX, UK, telephone - +44 1772 855723, facsimile - +44 1772 855715, e-mail - nicola.herbert@baesystems.com

Nicola Herbert is a Senior Systems and Software Engineer at BAE Systems Military Air Solutions, Warton. Nicola graduated from Lancaster University in 1997 with an honours degree in Computer Science and Software Engineering, before joining BAE Systems. Nicola spent five years working as a software engineer on various projects before moving into her current role in the Systems and Software Safety Group. In this role, she has provided specialist support to various aircraft projects in the area of software safety, including the assessment of flight-critical software. Nicola is currently working on methods of safety analysis for autonomous systems as part of her academic studies for the MSC at the University of York, and as part of the SEAS DTC.