Standard Practice for
Ensuring Dependability of Software Used in Unmanned Aircraft Systems (UAS)[1]

1. Scope

1.1 This standard practice intends to ensure the dependability of UAS software. Dependability includes both the safety and security aspects of the software.

1.2 This practice will focus on the following areas: (*a*) Organizational controls (for example, management, training) in place during software development. (*b*) Use of the software in the system, including its architecture and contribution to overall system safety and security. (*c*) Metrics and design analysis related to assessing the code. (*d*) Techniques and tools related to code review. (*e*) Quality assurance. (*f*) Testing of the software.

1.3 There is interest from industry and some parts of the CAAs to pursue an alternate means of compliance for software assurance for small UAS (sUAS).

1.4 This practice is intended to support sUAS operations. It is assumed that the risk of sUAS will vary based on concept of operations, environment, and other variables. The fact that there are no souls onboard the UAS may reduce or eliminate some hazards and risks. However, at the discretion of the CAA, this practice may be applied to other UAS operations.

1.5 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.*

2. Referenced Documents

2.1 *FAA Standard:*[2]

FAA 23.1309–1E System Safety Analysis and Assessment for Part 23 Airplanes

2.2 *IEC Standard:*[3]

IEC 62304 Medical Device Software—Software Life Cycle Processes

2.3 *ISO Standards:*[4]

ISO 9001 Quality Management Systems—Requirements

2.4 *ICAO Standard:*[5]

ICAO 9859 Safety Management Manual

2.5 *NASA Standard:*[6]

NASA Technical Briefs Making Sense out of SOUP (Software of Unknown Pedigree)

2.6 *RTCA Standards:*[7]

RTCA DO-178C Software Considerations in Airborne Systems and Equipment Certification

RTCA DO–278A Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems

RTCA DO-326 Airworthiness Security Process Specification

2.7 *Military Standards:*[8]

Department of Defense Joint Software System Safety Handbook

MIL-STD-882E Department of Defense Standard for System Safety

3. Terminology

3.1 *Definitions of Terms Specific to This Standard:*

3.1.1 *application programming interface (API)*—definition of the inputs and outputs for operations intended for use by other software modules.

3.1.2 *architecture*—architecture is made up of the definition of the sUAS Software components, the data that flows between

[1] This practice is under the jurisdiction of ASTM Committee F38 on Unmanned Aircraft Systems and is the direct responsibility of Subcommittee F38.01 on Airworthiness.

Current edition approved Sept. 1, 2016. Published September 2016. DOI: 10.1520/F3201-16.

[2] Available from Federal Aviation Administration (FAA), 800 Independence Ave., SW, Washington, DC 20591, http://www.faa.gov.

[3] Available from International Electrotechnical Commission (IEC), 3, rue de Varembé, P.O. Box 131, 1211 Geneva 20, Switzerland, http://www.iec.ch.

[4] Available from International Organization for Standardization (ISO), ISO Central Secretariat, BIBC II, Chemin de Blandonnet 8, CP 401, 1214 Vernier, Geneva, Switzerland, http://www.iso.org.

[5] Available from International Civil Aviation Organization (ICAO), 999 Robert-Bourassa Blvd., Montreal, Quebec H3C 5H7, Canada, http://www.icao.int.

[6] Available from U.S. National Air and Space Administration (NASA), 300 E. Street, SW, Suite 5R30, Washington, DC 20546, http://www.nasa.gov.

[7] Available from Radio Technical Commission for Aeronautics (RTCA), 1150 18th St., NW, Suite 910, Washington, DC 20036, http://www.rtca.org.

[8] Available from DLA Document Services, Building 4/D, 700 Robbins Ave., Philadelphia, PA 19111-5094, http://quicksearch.dla.mil.

the components (data flow), and the order of execution of the components (control flow).

3.1.3 *code churn*—the quantity and frequency of additions, deletions, and modifications to the source code for software.

3.1.4 *code coverage*—a measure used to describe the degree to which the source code of a program is tested by a particular test suite.

3.1.5 *customer*—includes stakeholders outside of the sUAS manufacturer who interface with the sUAS.

3.1.6 *dependability*—attribute of the software code that produces the consequences for which it was written, without adverse effects, in its intended environment.

3.1.7 *dynamic program analysis*—the practice of analyzing software while it is executing, for example monitoring memory access, allocation, and deallocation during program execution. For example, Valgrind is a popular open-source tool that performs this type of analysis.

3.1.8 *externally developed software (EDS)*—software developed outside of the sUAS manufacturer for which adequate records of the development process may not be available.

3.1.9 *EDS quality plan*—a plan to address the software quality in the event that EDS source code is not available. See Appendix X2 for more details.

3.1.10 *fuzz testing*—a testing technique wherein the input to a unit under test is unexpected in some way. Examples include testing with input that is invalid, unexpected, or random.

3.1.11 *internal user*—includes stakeholders within the sUAS manufacturer's organization who interface with the sUAS.

3.1.12 *internally developed software (IDS)*—software developed within the sUAS manufacturer's organization.

3.1.13 *penetration testing*—a testing method intended to identify and correct vulnerabilities and security defects by attempting to break, bypass, or tamper with software security controls.

3.1.14 *publish*—formalized release of a document to appropriate parties. A history should be maintained for published documents. The history may be part of revision control system, printed papers in a binder, or any other auditable system.

3.1.15 *quality assurance*—the practice of internally monitoring or auditing the development process.

3.1.16 *red team evaluation*—a process designed to detect network and system vulnerabilities and test security by taking an attacker-like approach to system, network, or data access, or combinations thereof.

3.1.17 *shall versus should versus may*—use of the word "shall" implies that a procedure or statement is mandatory and must be followed to comply with this practice, "should" implies recommended, and "may" implies optional at the discretion of the supplier, manufacturer, or operator. Since "shall" statements are requirements, they include sufficient detail needed to define compliance (for example, threshold values, test methods, oversight, and references to other standards). "Should" statements also represent parameters that could be used in safety evaluations, and could lead to development of future requirements. "May" statements are provided to clarify acceptability of a specific item or practice, and offer options for satisfying requirements.

3.1.18 *small unmanned aircraft system (sUAS)*—composed of small unmanned aircraft (sUA-see 4.2) and all required on-board subsystems, payload, control station, other required off-board subsystems, any required launch and recovery equipment, all required crew members, and command and control (C2) links between sUA and the control station.

3.1.19 *sUAS manufacturer*—the organization and personnel with design responsibility for the sUAS, including the dependability of the system software.

3.1.20 *sUAS Software*—includes both IDS and EDS.

3.1.21 *software baseline*—a known state of product software that has been formally reviewed and agreed on, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures.

3.1.22 *software vulnerability*—a mistake in software (also known as a weakness) that can be directly exploited to get a cyber-enabled capability to function in an unintended manner. Typically this is the violation of a reasonable security policy for the cyber-enabled capability resulting in a negative technical impact. Although all vulnerabilities involve a weakness, not all weaknesses are vulnerabilities. For example, Common Vulnerabilities and Exposures is a dictionary of common names for publicly known software-related vulnerabilities.

3.1.23 *statement coverage*—a testing technique that involves the execution of all the statements at least once in the source code. As a metric, it is used to calculate and measure the number of statements in the source code which have been executed.

3.1.24 *threat modeling*—a structured approach that enables the sUAS manufacturer to identify, quantify, and address the security risks associated with an application. The process involves systematically identifying security threats and rating them according to severity and level of occurrence probability. The overall goal for threat modeling (also known as attack modeling) is the creation of customized knowledge about potential attacks relevant to the application or organization. This customized knowledge guides decisions about changes to the code and security controls to implement.

3.1.25 *tier 1 requirements*—required tasks and activities in this practice for a software malfunction or penetration that would result in a slight reduction in sUAS functional capabilities or safety margins (for reference see Minor failure conditions per AC 23.1309–1E).

3.1.26 *tier 2 requirements*—required tasks and activities in this practice for any software malfunction or penetration that would result in a significant reduction in sUAS functional capabilities or safety margins with potential for injury (for reference see Major failure conditions per AC 23.1309–1E).

3.1.27 *tier 3 requirements*—required tasks and activities in this standard for any software malfunction or penetration that would result in a large reduction in sUAS functional capabilities or safety margins and could be expected to result in serious injury or fatality (for reference see Hazardous or more severe failure conditions per AC 23.1309–1E).

3.2 *Acronyms:*

3.2.1 *API*—Application Programming Interface

3.2.2 *CAA*—Civil Aviation Authority

3.2.3 *EDS*—Externally Developed Software

3.2.4 *FAA*—Federal Aviation Administration

3.2.5 *IDS*—Internally Developed Software

3.2.6 *sUA*—Small Unmanned Aircraft

3.2.7 *sUAS*—Small Unmanned Aircraft System

3.2.8 *UAS*—Unmanned Aircraft System

4. Applicability

4.1 The practice is written for all UAS intended for operation within airspace controlled by a CAA.

4.2 It is assumed that the maximum weight and airspeed of a sUAS will be specified by the nation's CAA. However, unless otherwise specified by a nation's CAA, this practice applies only to sUA that:

4.2.1 Have a maximum takeoff gross weight of 55 lb (25 kg) or less;

4.2.2 Have the capability to allow remote intervention by flight personnel in the management of the flight during normal operations.

4.3 This practice is intended for software that is part of a sUAS. It may be used by itself or in conjunction with other standards such as DO-178C, as deemed appropriate by the sUAS manufacturer in accordance with CAA guidance. This practice does not replace or supersede other standards, hence a sUAS manufacturer may choose to certify under alternatives such as DO-178C without reference to this practice, subject to CAA guidance. Appendix X1 contains guidance for producing artifacts corresponding to the requirements in Section 5.

4.4 The applicability of the practice extends to those software items in the sUAS that implement functions essential to safety. Software items that have no impact on safety are out of scope for this practice. For example, payload software on the sUAS that is not used to perform a safety-critical function is outside the scope of this practice.

5. Requirements

NOTE 1—The hazard analysis (see 5.2.1) will be used to determine the severity of a sUAS Software malfunction or failure and the corresponding tier. See Appendix X2 for examples of tier assignments to sUAS functions.

NOTE 2—The applicability of the each requirement is determined by the tier assignment and noted in parentheses next to the requirement. Unless otherwise indicated, sub-requirements inherit the tier assignments of the parent requirement (for example, if requirement 5.2.1 applies to Tiers 1, 2, and 3, then 5.2.1.1 also applies to the same tiers).

NOTE 3—Requirements may apply only to EDS, only to IDS, or to the sUAS Software (includes EDS and IDS). Unless otherwise indicated, sub-requirements apply to the kind of software (EDS, IDS, or sUAS Software) specified in the parent requirement. See Appendix X3 for scenarios for using this practice for EDS, IDS, and sUAS Software.

5.1 *Organizational Planning:*

5.1.1 *Tier 1, 2, 3*—The sUAS manufacturer shall publish an organizational software plan for sUAS Software.

5.1.1.1 This plan shall define the roles and responsibilities of each part of the manufacturer's organization involved in software acquisition, development, integration, and testing for all sUAS software projects in the organization.

5.1.1.2 The sUAS manufacturer should educate company executives and train employees on the risks and vulnerabilities of the EDS integration or software development approach, or both.

5.1.2 *Tier 2, 3*—The sUAS manufacturer shall record all uses and versions of EDS in the sUAS.

5.1.3 *Tier 3*—The sUAS manufacturer shall have an organizational response plan to address a flight critical software malfunction or penetration for sUAS Software.

5.1.3.1 The sUAS manufacturer should make information available to all users of its sUAS regarding the software issue and provide guidance within 24 hours of being made aware of the issue.

5.2 *sUAS Software Architecture and Use:*

5.2.1 *Tier 1, 2, 3*—The sUAS manufacturer shall conduct an analysis to determine the hazards and impacts associated with the potential malfunction, failure, or exploitation of the sUAS Software and identify potential risk mitigation.

5.2.1.1 The analysis shall define the sUAS Software's intended function(s) and document potential failure (gracefully or suddenly).

5.2.1.2 The analysis should be conducted using industry best practices (see references in Appendix X1) but should consider unique aspects of the sUAS size and operation.

5.2.1.3 Security vulnerabilities should be considered as possible causes of hazards in performing the hazard analysis.

5.2.2 *Tier 2, 3*—The sUAS manufacturer shall publish an EDS integration plan.

5.2.2.1 The plan shall document the tasks and milestones that need to be performed to acquire and integrate the EDS.

5.2.2.2 The plan should include release gates/checkpoints/milestones and associated criteria at one or more points during the acquisition and integration process, as well as configuration management for all EDS code and documents.

5.2.2.3 The EDS integration plan may be part of a larger software integration plan or other lifecycle documentation.

5.2.2.4 The EDS integration plan should address how configuration tables, data, and libraries that may be included in the EDS are integrated.

5.2.2.5 The sUAS manufacturer shall ensure that the EDS integration plan is folowed and track exceptions to the plan.

5.2.2.6 All exceptions to the EDS integration plan shall be incorporated into the plan and published.

5.2.3 *Tier 1, 2, 3*—The sUAS manufacturer shall publish a software development and integration plan for IDS.

5.2.3.1 The IDS development and integration plan shall establish the software baseline and document the tasks and milestones that need to be performed to develop the software for a specific project.

5.2.3.2 The IDS development and integration plan should include release gates/checkpoints/milestones and associated criteria at one or more points in the development process, and configuration management of code and documents.