A NEW METHODOLOGY FOR SOFTWARE RELIABILITY AND SAFETY ASSURANCE IN ATM SYSTEMS

Daniela Dell'Amura, Francesca Matarese
SESM – Sistemi Evoluti per la Sistemistica e i Modelli, A Finmeccanica Company
Via Circumvallazione Esterna - Zona ASI – 80014 Giugliano in Campania (Napoli - Italy)
www.sesm.it

Keywords: *safety, reliability, SWAL, methodology, software*

Abstract

The aim of this paper is to define a new methodology and to establish a future reference against which to assess software systems of ATM ground segments. The approach elaborated relies on the analysis of best practices both from other domains using dedicated standards and also from ANS, using the feedbacks of ATS providers.

An increasing proportion of Air Navigation System (ANS) functions is implemented by software and these functions are becoming more and more safety-critical. It is therefore necessary to define guidance on how assurance on reliability may be provided for software.

However today, no ANS software-related standard exists which neither fulfils ANS specificities (especially for ground part of ANS), nor is widely spread and extensively used by ANS community (at least not enough to become a de facto standard).

The only methodology and assurance level proposed for ANS systems, which is not a standard yet, is the one of EUROCONTROL and EUROCAE, who defined the Software Assurance Level (SWAL) and provided ED-153 for recommendations and requirements on the major processes necessary to provide safety assurance for software in ANS systems. ED-153 can be applied whenever it is possible to assess the whole software lifecycle (from the design phase), while most of existing ATM systems are the result of an integration between new software products and old ones, for whom is no more possible to assess the first steps of lifecycle, but just their service history, based on problem reports opened during their operational life.

ANSPs require the assessment of safety impact of the introduction of new software components in existing systems and do not accept just the legally required certification of interoperability. To this purpose, we developed and proposed an innovative approach, based on the verification of SWAL for new safety components and of service history evidences for the old ones. The new methodology is a customization of Safety Assessment Methodology. It has been proposed to several ANSPs around Europe, who accepted and validated it.

1 General introduction

Introducing new technology into safety-critical environments can cause more problems than it solves if it is not done carefully.

In fact, an increasing proportion of Air Navigation Service (ANS) functions, in particular the ones related to ground segments, is implemented by software. These functions are becoming more and more safety-critical, as the provision of an ANS is inherently a risky operation, providing the primary means of avoiding aircraft collisions. Moreover, the introduction of new navigation systems highlights the need for efficient tools to assess the possible impact of these systems on the current safety levels.

It is necessary to define guidance on how assurance on safety may be provided for software, but today no ANS software-related standard exists, which neither fulfils ANS specificities (especially for ground part), nor is widely spread and extensively used by ANS community, at least not enough to become a de facto standard.

EUROCONTROL has suggested a Safety Assessment Methodology (SAM), which is not a standard but aims at defining practices to assure safety of an ANS system during its whole lifecycle. Its main limitation consists of not evaluating safety level of existing legacy systems, which have been developed over an extended period of time and for whom the only evidence that they are 'tolerably safe' is that they have proved themselves to be so over years of operation. [1]

Therefore, the aim of this paper is to define a new methodology and to establish a future reference against which to certify safety of software systems of ATM ground segments. This methodology assesses safety of new integrated systems, constituted by old legacy and new ones.

2 State of the art

2.1 Software reliability models

Safety of a system is defined as freedom from unacceptable risk, which is the combination of the overall probability of occurrence of a harmful effect, induced by a hazard, and the severity of that effect.

Severity is assessed by Air Navigation Service Providers (ANSPs), who know the consequences that can affect the overall system. The probability of occurrence instead can be assessed by stakeholders and be equated to reliability that is used to describe the probability of the system, operating in a given environment and within the designed range of input, without failure. Therefore, software reliability is defined as the probability that software will not cause a system failure, over a specified time period under specified conditions, and can be used to assess probability of occurrence of hazards related to existing legacy systems.

Unlike hardware reliability engineering, which was first introduced as a discipline during World War II, the software reliability is much younger, beginning in the mid 1970's, when the software development environment was reasonably stable. The known "bathtub" curve for Hardware Reliability does not apply to software, since software does not typically wear out. However, if the hardware life cycle is likened to the software development through deployment cycle, the curve can be analogous. The Software bathtub curve is shown in Fig. 1:

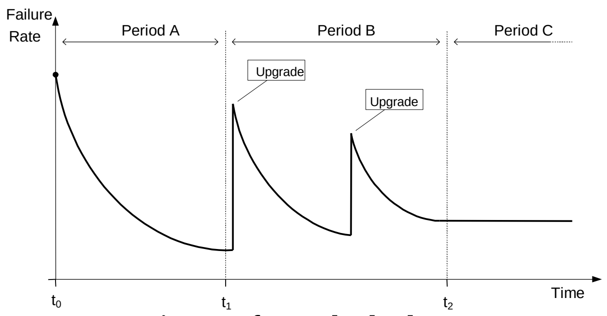

Fig. 1: Software bathtub curve

For software, the time points are defined as follows:

- t_0 is the time when testing begins. Period A (from t_0 to t_1) is considered to be the debug phase. Coding errors, more specifically errors found and corrected or operation not in compliance with the requirements specification, are identified and resolved. This is one key distinction between hardware and software reliability: the "clock" is different. Development/test time is not included in the hardware reliability calculation but is included for software.
- t_1 is the initial deployment (distribution) time. Failures occurring during Period B (from t_1 to t_2) are found either by users or through post deployment testing. For these errors, work-around or subsequent releases typically are issued (but not necessarily in direct correspondence to each error reported).
- t_2 is the time when the software reaches the end of its useful life. Most errors reported during Period C (after t_2) reflect the inability of the software to meet the changing needs of the customer. In this frame of reference, although the software is still functioning according to its original specification and is not

considered to have failed, that specification is no longer adequate for the current needs. The software has reached the end of its useful life (obsolescence) much like the wear out of a hardware item. Failures reported during Period C may be the basis for generating the requirements for a new system.

Usually hardware upgrades occur during Period A, when initial failures often identify required changes. Software upgrades, on the other hand, occur in both Periods A and B. Thus, the Period B line is not really flat for software but contains many mini-cycles of periods A and B: an upgrade occurs, most of the errors introduced during the upgrade are detected and removed, another upgrade occurs, etc.

Although the failure rate drops after each upgrade in Period B, it may not reach the initial level achieved at initial deployment. Since each upgrade represents a mini development cycle, modifications may introduce new defects in other parts of the software unrelated to the modification itself.

An upgrade often focuses on new requirements and its testing may not typically encompass the entire system. Additionally, the implementation of new requirements may inversely impact or be in conflict with the original design. The more upgrades that occur, the greater the likelihood that the overall system design will be compromised, increasing the potential for increased failure rate, and hence lower reliability. This scenario is now occurring in many legacy systems (as existing ANSs), which have recently entered Period C, triggering current reengineering efforts.[2]

2.2 Peculiarities of ANS software

ANSPs are responsible for ANSs they provide.[3] Whenever they need to upgrade the ATM system, they have to demonstrate to the National Supervision Authority (NSA) that it is still reliable and that it will not impact on existing safety level. To this aim, ANSPs ask to the stakeholders to provide safety assessment of the new system, composed by newly developed elements and already existing ones.

At the same time, when ANSPs receive the result of new system safety assessment, they have to evaluate it in the context of the already existing legacy system, assessing the resulting level of safety of the integrated system.

A system upgrade often focuses on new functionalities, whose implementation may inversely impact or be in conflict with the original system. The more upgrades that occur, the greater the likelihood that the overall system design will be compromised, increasing the potential for increased failure rate.

The limit of existing safety assessment methodologies is that they evaluate safety level of new subsystems as stand-alone, not in combination with existing legacy ones. This approach is not acceptable because ANSPs that decide to upgrade their existing ANS systems rarely change the overall system, but just a part of it. That means that stakeholders are in charge to ensure reliability of the “change” they are providing, ignoring possible new failures that could occur in the new integrated system.

Sometimes, it happens that existing sub-systems are assessed, but as black-boxes, to be tested just indirectly through tests on new sub-system functionalities. No additional tests are usually performed on old functionalities, which could on the contrary be affected by the new ones.

Moreover, when providing a new part of the system, this is composed by different sub-systems, some of them of new concept, others already developed. So it happens that two different kind of difficulties have to be faced by Safety Engineers: the one of evaluating safety level of the integration between old legacy software systems with new ones, and the one of the deployment of newly developed software components integrated with already existing ones.

Software reliability is defined as the probability that software will not cause a system failure and can be used to assess probability of occurrence of hazards, based on service history metrics for existing legacy systems, or on the quality of new subsystems.

To most software engineers, reliability is equated to correctness, which is the reliability of the delivered code is related to the quality of all the processes of software lifecycle. According to this definition, EUROCONTROL defined Software Assurance Level (SWAL) as a uniform measure of how the software was developed, transferred into operation, maintained and decommissioned and a measure of the ability of the product to function as intended.

3 Regulatory Framework

3.1 Software safety-oriented standards

Some safety-oriented standards to assess software reliability exist, such as ED12B/DO178B, ISO/IEC 12207, ED109, IEC 61508-3, ED12B/DO178B and CMMI, but which first requires to be tailored to a domain of application (this has not yet been done for ANS ground segments).

Here below a short description of these international standards:

- ISO/IEC 12207 - Information Technology - Software Engineering - Software Life Cycle Processes.
- ED109/DO278 - Guidelines for Communication, Navigation, Surveillance, and Air Traffic Management (CNS/ ATM) Systems Software Integrity Assurance.
- IEC 61508-3 - Functional safety of electrical/electronic/programmable electronic safety-related systems. Part 3: Software Requirements.
- ED12B/DO178B - Software Considerations in Airborne Systems and Equipment Certification.
- CMMI - Capability Maturity Model Integration.

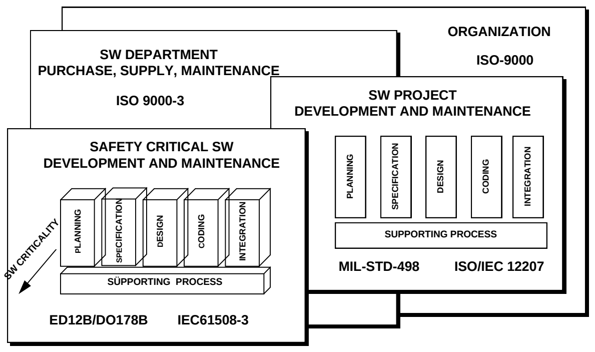

Fig. 2: Scope and Interrelationships of Standards

The ISO/IEC 12207 Standard is currently considered as reflecting the best practices for all processes and activities of a Software lifecycle.

The IEC 61508-3 and the ED12B/DO178B cover the lifecycle of safety critical software. The IEC 61508-3 is part of an emerging generic standard (IEC 61508) addressing the functional safety of safety-related systems (in particular of the Equipment Under control (EUC). This generic standard is expected to be tailored to a specific sector of application.

The EB12B/DO178B Standard defines recommended practices for the development of software in airborne systems and equipment. The Standard is not mandatory, but represents an international consensus in the avionics industry.

The MIL-STD-498 has been used in ANS industry. This standard is now superseded by the ISO/IEC 12207.

ED109/DO278 applies to software contained in CNS/ATM systems used in ground or space-based applications shown by a system safety assessment process to affect the safety of aircraft occupants or airframe in its operational environment. A description of the prerequisite safety assessment process is not included in ED109/DO278. ED109/DO278 is not intended to be a development standard nor a process document.

The CMMI is a model, whose purpose is:

- to provide some guidance for an organisation to improve its processes,
- to serve as a reference to assess process capability/maturity level of the organization, and then to benchmark organizations.

The scope of this model covers the development, acquisition, and maintenance of product or services. It may be used in various disciplines: System engineering, Software Engineering, Project Management and Supplier Sourcing. The extension to other disciplines (including safety engineering) is possible but requires a specific interpretation of the model to the discipline.[4]

3.2 ANS Software Safety Assessment

None of the previous standards is ANS software-related, neither fulfils ANS specificities (especially for ground part), nor is widely spread and extensively used by ANS community, at least not enough to become a de facto standard.

For this reason, EUROCONTROL has proposed a new approach, Recommendation for ANS Software, based on the reuse of IEC/ISO12207 processes structure, which has the widest coverage (from definition till decommissioning) of ANS needs, focusing on "ground" segment.

EUROCAE ED153 is derived from these EUROCONTROL Recommendations. ED153 is not a standard, but defines practices to assure safety of an ANS system during its whole lifecycle. It has been delivered to provide guidance on how to be compliant with EC Regulations on ATM Safety [5].

ED153 covers quality and safety related activities from the beginning of the system definition till decommissioning. Unfortunately it still appears to have some limitations in its applicability, due to the fact that it aims at assuring the safety and reliability of not yet available software.

The only methodology and assurance level proposed for ANS systems, which is not a standard yet, is the one of EUROCONTROL and EUROCAE, who defined the SWAL as part of Preliminary System Safety Assessment (PSSA) process, in the frame of SAM. A SWAL relies upon planned and systematic actions necessary to provide confidence and assurance (through arguments, evidences or other means) that a software product or process satisfies given requirements. SWAL is based upon the contribution of software to potential consequences of its anomalous behaviour as determined by the system safety assessment process. The SWAL implies that the level of effort recommended to showing compliance with Safety Requirements (SRs) varies with the severity of the end effect of the software failure and the probability/likelihood of occurrence of the end effect.

The SWAL is a uniform measure of how the software was developed, transferred into operation, maintained and decommissioned and a measure of the ability of the product to function as intended. [4][6]

ED153 can be applied whenever it is possible to assess the whole software lifecycle (from the design phase), while most of existing ATM systems are the result of an integration between new software products and old ones, for whom it is no more possible to assess the first steps of lifecycle, but just their service history, based on problem reports opened during their operational life.

4 Methodology

The ANS SAM has been developed to reflect best practices for safety assessment of ANSs and to provide guidance for their application.

SAM describes a generic process for the safety assessment of ANSs. It covers the complete life cycle of the ANS system, from initial planning and system definition to de-commissioning.

ANS SAM methodology provides Guidance Material on how to assess what is a "change". Safety management practice demands that, before making a change to a safety related system, appropriate steps to ensure that the change does not introduce an unacceptable risk into the system has been taken into account. Therefore, a simple hazard identification procedure is requested to ANSPs by NSAs to determine whether it is necessary to re-assess the system safety level.

To be able to answer ANSPs requirement of validation of the whole new operating integrated system, an innovative approach has been proposed, based on the verification of

SWAL for new software components and of service history evidences collection for the old ones. The new methodology is therefore a customization of EUROCONTROL's SAM.

4.1 Analyses

4.1.1 Fault Tree Analysis (FTA)

The FTA is performed starting from the Functional Hazard Assessment (FHA), provided by ANSPs, through which it is possible to identify credible system hazards and to classify them according to their severity.

A fault tree is developed for each Top Event identified. A fault tree is a model that graphically and logically represents the various combinations of possible failures and events occurring in a system that lead to a failure condition at the top.

Once the FTA is performed, starting from the probability assigned to the Safety Objective, the probability to be assigned to each element in the diagram is determined by applying a top down process. In this way, it is possible to apportion the requirements coming from the Safety Objectives (SOs) to physical components functionalities, thus allowing a direct link of these requirements to the physical components failures that affect these functions, by performing a dedicated FMECA.

4.1.2 Failure Mode Effects and Criticality Analysis (FMECA)

FMECA is carried out on physical software components in order to identify possible failure modes, their effects at different levels, their connection to FTA, their severity, their possible mitigation means and the resulting new SRs after mitigation.

Here below the representation of Risk Classification Scheme, with qualitative and quantitative ranges, used for evaluating the risk associated to the Failure Modes, that has to be at least tolerable.[7]

		PROBABILITY OF OCCURRENCE				
		Extr. Rare	Rare	Occa-sional	Frequent	Very Frequent
		$< 10^{-7}$	from 10^{-7} to 10^{-5}	from 10^{-5} to 10^{-3}	from 10^{-3} to 10^{-1}	$>10^{-1}$
SEVERITY CLASS	I Accident	🟨	🟥	🟥	🟥	🟥
	II Serious Incident	🟩	🟨	🟥	🟥	🟥
	III Major Incident	🟩	🟩	🟨	🟥	🟥
	IV Significant Incident	🟩	🟩	🟩	🟨	🟥
	V No Safety Effect	🟩	🟩	🟩	🟩	🟨

🟩 Acceptable 🟨 Tolerable 🟥 Unacceptable

Table 1: Risk Classification Scheme

4.1.3 Safety Requirements and SWAL allocation

FTA is performed in order to determine the SRs; this is done by deriving a functional breakdown that allows apportioning the requirements coming from the SOs to physical components functionalities, thus showing a direct link of these requirements to the physical components failures that affect these functions, by performing a dedicated FMECA.

FMECA allows identifying connections between failure modes of system components and SRs. In order to calculate SRs after a Mitigation Mean is implemented, it is necessary to consider the connection between FTA and FMECA, i.e. between Basic Events and Failure Modes.

After having determined SRs it is possible to translate them into SWAL objectives for software component functionality.

To allocate a SWAL to an ATM software function, the likelihood that, once software fails, this software failure can generate an end effect, which has a certain severity, is identified. That couple (severity, likelihood) corresponds to a certain SWAL, according to the following matrix:

Severity / **Likelihood (Pe x Ph)**	**1**	**2**	**3**	**4**	**5**
V.Frequent	SWAL1	SWAL2	SWAL3	SWAL4	SWAL4
Frequent	SWAL2	SWAL3	SWAL3	SWAL4	SWAL4
Occasional	SWAL3	SWAL3	SWAL4	SWAL4	SWAL4
Rare	SWAL4	SWAL4	SWAL4	SWAL4	SWAL4
Extr.Rare	SWAL4	SWAL4	SWAL4	SWAL4	SWAL4

Table 2: SWAL matrix

SWAL allocation is possible only for new designed software; for already existing software, SRs, which correspond to a certain range of acceptable likelihood, have to be demonstrated by service history.

4.2 Collection of evidences of compliance

System Safety Assessment (SSA) process aims at demonstrating that the system as implemented achieves an acceptable (or at least a tolerable) risk and consequently satisfies its Safety Objectives specified in the FHA and the system elements meet their Safety Requirements specified in the PSSA.

The compliance to each analysed SR implies the compliance to each SO. In order to demonstrate system compliance with SRs, two options can be considered: Service History Analysis or SWAL assessment.

As already explained in §2.2, existing sub-systems are usually not considered in safety assessment of new ANS systems that integrate them, or rather they are assessed as black boxes.

The new proposed methodology requires that each element of existing legacy subsystem is considered as part of the new integrated system.

Service History Analysis provides evidences of reliability for those software components whose history data are available, resulting in the evidence of their reliability in the past.

The SRs are expressed in terms of Failure Rates. These requirements are then compared with the Failure Rates resulting from service history analysis, to prove compliance.

In order to calculate the Failure Rate associated to each software component involved in the analysis, the following parameters have been evaluated:

- <u>Number of operative hours in the considered period of time calculation.</u>
- <u>Overall number of operative hours in the considered period of time calculation</u> (the number of operative hours in the considered period of time per the number of sites in which the specific software component is installed).
- <u>Number of Failures per CSCI reported.</u>
- <u>Failure Rate calculation</u> (the number of failures occurred in the considered period of time is divided by the number of operative hours; this result represents the Failure Rate, the number of Failures per unit of operative hour).

SWALs instead, are designed to provide a level of confidence that the software will be developed and can be integrated in the equipment and then in the system in order to manage risks due to software failure.

The way to provide this level of confidence and assurance is by defining some objectives that will satisfy this level of assurance.

These objectives address the software acquisition, development, integration, maintenance, operation, and all processes of the software lifecycle and identify what is to be done to satisfy a level of assurance. These objectives intend to give confidence that the assurance level is satisfied by showing evidences.

These evidences are produced by activities, which achieve these objectives. Therefore, in order to provide evidences that such activities have been correctly performed it is necessary to produce logs about all software lifecycle phases.[8]

5 Conclusions

ANSPs require the assessment of safety impact of the introduction of new software components in existing legacy systems. No standard or guidance material exists for evaluating this complex situation.

To this purpose, we developed and proposed an innovative approach, based on the verification of SWAL for new safety components and of service history evidences for the old ones. Both evidences collection methods are necessary to give assurance that the ANS software answers to SRs, because in case of new systems it is possible to evaluate the whole lifecycle activities but no service history exists; vice versa in case of existing ANS software.

The new methodology is a customization of EUROCONTROL's SAM. It has been proposed to several ANSPs around Europe (Italy, Luxembourg, Cyprus, Malta, Georgia, Turkey, Romania) who accepted and validated it.

6 References

[1] EUROCONTROL, *Air Navigation System Safety Assessment Methodology*, Ed. 2.1, 2006.

[2] J. Marciniak, R. Vienneau, *Software Engineering Baselines*, 1996.

[3] COMMISSION REGULATION (EC) No 2096/2005 of 20 December 2005 laying down common requirements for the provision of air navigation services.

[4] EUROCONTROL, *Recommendations For A.N.S Software*, Ed. 1.0, 2005.

[5] COMMISSION REGULATION (EC) No 482/2008 of 30 May 2008 establishing a software safety assurance system to be implemented by air navigation service providers and amending Annex II to Regulation (EC) No 2096/2005.

[6] EUROCAE, *Guidelines for ANS Software Safety Assurance*, Ed. 1.0, 2009.

[7] EUROCONTROL, *EUROCONTROL Safety Regulatory Requirement ESARR4 Risk Assessment and Mitigation in ATM*, Ed. 1.0, 2001.

[8] EUROCONTROL, *EUROCONTROL Safety Regulatory Requirement ESARR6 Software In ATM Systems*, Ed. 1.0, 2003.

7 Contact Author Email Address

Contact authors email addresses:
Daniela Dell'Amura ddellamura@sesm.it
Francesca Matarese fmatarese@sesm.it

Copyright Statement