Safety-Critical Certification – Options for Your Next Device

ABSTRACT:

During a recent Embedded Systems Conference's annual Boston show, Micriμm announced its strategic alliance with Validated Software Corporation (VSC). In the press release, the reason for the alliance is "The goal of the alliance is to advance their products as the low-risk, cost-effective alternative to other commercial and in-house systems. With strategic and tactical alignment on designs, technologies, and accounts, the partnership will greatly increase customer access to proven, cost effective safety-critical compliant RTOS technology."

The purpose of this paper is to present:

- A brief overview of safety-critical software standards;
- Market trends;
- VSC's current support for Micriμm real-time products.

AUDIENCE:

Medical, Industrial, Transportation, and Avionics device developers and managers who are considering the use of a commercial embedded RTOS as the foundation for their development.

1902 Wright Place, Suite 200 • Carlsbad, California 92009 • (760) 531-5290

Table of Contents

Safety-Critical Certification - Options for Your Next Device

Safety-critical Software Standards

With the growing use of embedded computers in everyday items comes the increasing probability that the software one develops will ultimately intersect with a safety case. Therefore, many developers who have not been exposed to software safety standards have to develop to them now. However, there are a number of ways that developers can reduce both the risk and expense of writing safe software. One way is to eliminate all non-core development activity, such as RTOS development, and give it to the experts. This reduces the total burden, and ensures that “experts” are managing that part of your software program. While the line item expense for licensing an RTOS may stand out, a review of what goes into the development, maintenance, and support provided by a responsible software vendor shows that utilizing their services is clearly cost effective. The same can be said for the vendor who validates an RTOS to various safety standards. While safety-critical software development will never be considered “cheap,” it can be competitive and cost effective.

There are a number of definitions for “safety-critical.” According to Wikipedia, “A safety-critical system, subsystem, condition, event, operation, process or item is one whose proper recognition, control, performance or tolerance is essential to system operation such that it does not jeopardize public safety."

A simpler definition of a safety-critical system is any system whose failure:

- Causes or allows harm or death to humans and their pets;
- Causes or allows environmental damage;
- Causes or allows the loss of large capital equipment.

What are Safety-critical Software Standards?

Software development standards are practices and methodologies that consistently produce robust, reliable software. Safety-critical software development standards are used to develop robust reliable software for systems whose failure will lead to unacceptably adverse ends.

While not complete, the list below includes a majority of the dominant internationally recognized standards.

- RTCA DO-178C, DO-278A, EUROCAE ED-12C, ED-109A – Avionics (Airborne)
- IEC-61508 – Industrial control, automation, manufacturing, safety systems
- UL-1998 – Home, consumer, industrial safety
- IEC-62304 – Medical, clinical and home diagnostic and therapy
- FDA/CDRH 510(k) – Medical clinical and home diagnostic and therapy
- CENELEC EN-50128 – Rail/transportation, locomotive, signaling, and control

At a high level, most safety-critical standards have two areas of application. The first area involves detailed knowledge of the problem domain itself i.e., its technology, techniques, theory of operation, etc. An example would be the research, techniques, and theory that comprise a new heart/lung machine. The second area involves detailed knowledge of the "implementation" side of the device, i.e. the system and software domain.

The second part of the standard usually provides guidance for the "implementers", those who create the physical system from a set of requirements. It is this second area that most software developers would recognize. Most safety-critical standards provide a great deal of guidance for those who implement the actual system. From a software development perspective there are three components provided by the standard directly or as a by-product of the guidance given to the domain experts:

- Requirements – as derived from the experts in the device domain;
- The level of criticality of the software – as established by analysis within the guidance of the standard;
- The software lifecycle – as defined by the standard.

Domain Specific

The domain or device specific portion of the safety standard generally provides guidance on:

- Identification of hazards
- Definition of hazards
- Scope of hazards
- Probability of a hazard manifesting
- Human factor analysis
- Scope of failure
- Probabilities of failure
- Consequences of failure
- Severity of failure
- Techniques to reduce and remediation of failures
- Risk analysis and abatement
- Assessment of risk
- Tolerance for risk
- Methods, practices, procedures, planning, etc. that pertain to the development of safety systems.

It is the domain experts that provide the knowledge i.e., the requirements, hazard analysis, etc. that the "implementers" utilize to design the physical system.

The above is not meant to be an exhaustive list, nor is it specific to a particular market, standard, or area of application (domain). It is provided to clarify the rationale for the use of so many different standards.

Non-Domain Specific

The second part of the standard is tied to what most software developers would recognize as the software development lifecycle. As stated earlier, the two most critical derivatives of the domain specific portion of the safety-critical standard are the requirements and the level of criticality of the software i.e., what is being built and the degree of rigor necessary to insure that it is built correctly. It is here that the teams responsible for the hardware and software design begin their development process. Additionally, it is at this stage that a great number of similarities between safety-critical software standards become noticeable.

A brief look at a cross section of the safety-critical software standards used in Avionics, Transportation, Medical and Industrial applications shows a great deal of overlap between them. In fact, as one person recently pointed out during a webinar on the topic of safety-critical software development, at a high level there is very little that differentiates good software engineering practices and safety-critical software development practices. It is not surprising. As in life, many paths lead to a single destination, but only a few are "optimal" ones.

Regardless of the specific methods, paradigms, processes and procedures, most developers agree on the five major components that make up the software life cycle.

1. Requirements Analysis and Definition
2. Software Design
3. Implementation and Unit Testing
4. Integration and System Testing
5. Operation and Maintenance

All safety-critical development standards additionally include a "**Planning**" step. It is “Step 0” of safety-critical development.

When one talks about the software life cycle, the word "Plan" is taken to be both a noun and a verb. In order to create reliable software one must “plan”, as well as have a “plan”.

Plan (noun): a specific project or definite purpose, a scheme or method of acting, doing, and proceeding.

Plan (verb): to make plans, to arrange a method or scheme beforehand for any work. To plan ahead.

An additional item that separates traditional software development from Safety-critical Software Development practices is “**Process**”. For those who are not familiar with safety-critical development practices, “Process” is the bedrock of Safety-Critical Software. Without process, interjecting consistency (of a proven method) to the essential components and activities of safety-critical development there is no assurance that success or, in this case safety, could be maintained. Yes, process is considered a necessary component in commercial software development; it is the *degree* of process that differentiates safety-critical from commercial development practices.

Criticality

Most if not all standards provide for varying degrees of "Criticality" (a relative measure of severity or risk) that imply different levels of rigor that have to be applied to the development process and software in order to comply with the standard. A common, yet not entirely correct, view of criticality is that is it a measure of the harm that might result from a software failure. In reality, the definition of criticality differs considerably between standards.

One measure of criticality, termed the "Safety Integrity Level" (SIL) is used by several standards. For European-based IEC standards, a simplistic definition of the SIL of a safety function is the measure of its risk reduction. A SIL of “1” is the lowest amount of risk reduction and a SIL of “4” is the highest. Standards outside of the IEC use the SIL acronym but it may have a slightly different meaning. In Avionics, the criticality is expressed as "level" with "Level A”, being the most stringent i.e. a failure considered having catastrophic potential. A failure of a "Level E" system has zero impact on the operation of the aircraft.

The criticality of a system is governed by the domain specific portion of a safety standard; it may or may not have dependencies on hardware aspects of the system. For the standard IEC-61508, the SIL of a device is derived from both the risk reduction gained by hardware and the software system as a whole. Other standards view the criticality of hardware and software separately.

A software developer only needs to know two things in order to develop software that complies with a standard. They have to understand the standard and its criticality. With this knowledge in hand, a software developer simply adheres to the development practices, procedures, methodologies, etc. dictated by the standard and the device criticality. While overly simplified, this is the same concept that enables commercial software and certification companies such as Micriμm and Validated Software Corporation (VSC) to provide off the shelf software and validation products that serve a great number of safety-critical industries. To accomplish this Micriμm and VSC take advantage of some aspects of each standard’s criticality rankings and apply flexibility in utilizing different techniques to comply with the requirements of a standard.

While it is beyond the scope of this paper to detail the internal requirements of the safety-critical standards, a brief clarification is in order. Rather than develop a version of software and its supporting evidence of compliance for each criticality level of each standard, both Micriμm’s software and VSC’s certification products exploit the fact that all standards allow one to surpass its requirements. In other words, they develop their products so that they comply with the highest levels that can be achieved and use that to satisfy lower criticality requirements. This is allowed WITHIN the different levels of criticality of a standard.

The second area of flexibility that VSC and Micriμm exploit is flexibility BETWEEN the different standards --they utilize the best of class safety techniques offered by individual standards. This serves two purposes. As mentioned earlier, it brings a higher overall safety case to bear on most of the requirements of any particular standard, and it allows a great deal of reuse of compliance evidence between the various standards.

For a customer, that means that even if they are developing a low-level criticality i.e., SIL 1, or Level E, they know that the core RTOS software they received from Micriμm can be certified to the highest levels of criticality. Furthermore, their certification package utilizes the best of the best techniques offered by the cross section of standards that VSC supports.

While not entirely germane to our topic, it is the ability to develop once and retest on a per-hardware target basis that allows VSC to provide such cost effective pricing on its certification products. As will be covered later, the core validation components for each standard represent a huge investment of time and labor. However, due to μC/OS’ small and efficient footprint, porting and testing to individual hardware targets is labor intensive but not nearly to the degree imposed by other real-time operating systems.

Safety-critical Software Costs

As should be obvious from the previous discussion, Safety-critical Software Development is an expensive proposition. To put it in perspective, the industry average for **commercial embedded software** is typically between $15 and $30 per line of code (LOC). It may sound expensive but

this is not only the expense of coding the software; this figure includes architecture and design, implementation, testing and release. As a rule of thumb, safety-critical software is 5 to 10 times as expensive to develop. It ranges from $75 to $150 per SLOC.

A high-level description of the Safety-critical Development process involves:

- Process, methodology & documentation
    - Planning
    - Development
    - Requirements
    - Verification
    - Configuration management
    - Quality assurance
- Code
    - Source code
    - Test code
    - Test scripts
    - Build & object code
- Test
    - Code coverage and analysis
    - Unit/white-box
    - integration/black-box
    - Structural Analysis
    - Plan for tool usage
- Results
    - Accomplishment summary
    - Unit & integration test
    - Requirements traceability
    - Trace matrix

To put this in perspective, the space shuttle's software, arguably the world's best, cost roughly $1,200 per each of its 420,000 SLOC.

Trends

Safety-critical software has become a very important topic to an ever-growing group of developers, manufacturers, and users. Why? It is no secret that digital control is playing a greater role in almost every human activity. In manufacturing, for example, computer driven devices are more accurate, faster, less mistake prone, and essentially tireless in their efforts. One often hears that the cost of computers and digital electronics in automobiles is nearing 50% of the total manufacturing cost of an automobile. Perhaps these statistics put it in perspective with the

following metrics: a typical GM car in 1970 had approximately 100,000 SLOC, in 1990, it had approximately 1,000,000 SLOC, and in 2010, it has approximately 100,000,000 SLOC.

While much of the software contributing to this exponential growth is not safety or mission critical, a significant and growing percentage is. Device complexity is growing and much of it is due to convergence. It is clear that single-function safety-critical devices are going the way of the dodo bird. Example: defibrillators used to be standalone devices, i.e., all they did was monitor the heart rhythm and deliver a metered, therapeutic dose of electrical energy to the heart. However, like many digital devices, today's defibrillators also provide wireless connectivity, file systems, USB, and other infrastructure software.

Another example, one that is not new but is really starting to accelerate, is demand for consumer electronics-like user experiences in the clinical setting. One physician recent confided that his car stereo had a more intuitive and "fun to use" user interfaces than his $200,000 diagnostic device.

For other devices, the addition of things like wireless connectivity is essential. It has the ability to bring experts to the scene of an event and share information for, excuse the term, "post mortem" analysis. Developers are being asked to add new features to their products, but the funding for travel, training or additional staff has been reduced or eliminated. As a result, development efforts are starting to slip. The statistics do not lie. Respondents to a 2009 *TechInsights* survey claimed that 57% of their projects were completed late, on average finishing 4.4 months behind schedule.

Working in an industry that develops safety-critical products is no protection against the economic conditions ravaging the economy. Some industries are more sensitive to economic fluctuations than others are. One example would be private aircraft manufacturing. The impact is not limited to small aircraft manufacturers; it trickles down to the instrument, radio, ground-based equipment and facilities, etc. So, like everyone else, safety-critical developers are being asked to do more with less and cut every dime of expense. For developers in medicine, industry, avionics, and transportation this could not have come at a worse time. For, unlike their brethren developing commercial grade software, safety-critical developers who cut corners, or hack a bug fix, may have to live with dire consequences of their actions.

Developers have Options

The addition of an RTOS brings many benefits to the developer. Portability, reusability and a superior programming model are just a few of the immediate benefits. The addition of major feature sets, such as networking and connectivity stacks, graphical user interfaces, and file systems, can be added with little difficulty to help offset feature-based demands from the field. However, the addition of an RTOS kernel, along with advanced software stacks, also has the potential to drive its certification costs beyond the budget at which a manufacturer can make a reasonable return on their investment.

This is where VSC and Micriμm fill the void. Micriμm with VSC's backing deliver integrated networking, USB, and fault-tolerant file systems, and deliver them with complete certification support. Furthermore, Micriμm has made an art of working with semiconductor vendors to make μC/OS available to their customers. With VSC's certification support added to the mix, medical, avionics, industrial and transportation application developers are able to order certifications as easily as they order their hardware and software, and to do so with the knowledge that it will arrive ready to submit to any of the major regulatory agencies around the globe.

Meet Validated Software and Micriμm

Over the last decade, Micriμm and Validated Software (VSC) have worked together to provide cost effective safety-critical software and its supporting certification artifacts so that their

customers did not have to divert their focus from the features that differentiate their products from their competitors. With their customer-centric orientation, both companies have cultivated a reputation for delivering high-value products without any fuss or hassles. Developed specifically for resource constrained embedded devices, Micriμm's μC/OS embedded components are used in millions of devices worldwide, and include numerous designs in building and industrial control and automation, medical devices, avionics, rail and transportation devices, and a variety of other mission, and safety-critical applications. VSC compliments Micriμm's efficient RTOS code with its semi-custom, certification ready Validation Suites™ and "off the shelf" Validation Kits™. VSC's stated mission is to reduce risk; they do so with a practical and comprehensive approach to developing and delivering their validation products with an unheard of *guarantee* that the components they support will receive certification.

In today's economy, companies can no longer pay a premium without receiving equal value in return. With Micriμm and VSC announcing their formal commitment to a strategic alliance, customers of both companies can expect to receive faster service, greater alignment on roadmaps, and better service due to increased scale.

Micriμm

Micriμm is a leader in embedded software components. The company's flagship μC/OS family is recognized for a variety of features and benefits including unparalleled reliability, performance, dependability, impeccable source code, and vast documentation. In addition, several Micriμm modules contain certifications that meet the rigorous safety-critical standards demanded by several industries. Both Micriμm founder Jean Labrosse and Micriμm vice-president Christian Legare are widely known and respected as industry experts based upon their broad visibility within the industry. In addition to the educational tracts they deliver at Embedded System Conferences and other industry events, Jean's books on RTOS are the most popular books on the subject, and have been used by most embedded system developers since their college days.

Micriμm products are created for engineers by engineers. If you are new to Micriμm, contact them to discuss your project requirements, challenges, and goals. Let them show you how their commercial RTOS family provides time-to-market advantages at a price you can afford. Micriμm will show you how their software module advantages translate into substantial time and resource savings today and in the future.

Validated Software

VSC has been developing embedded safety-critical software and its supporting evidence of standards compliance (certification artifacts) for over ten years. VSC supports every commercial 16 and 32-bit processor and toolset that μC/OS supports, and VSC's core competency is providing certification support for embedded COTS RTOS components. Not only is VSC uniquely qualified to provide certification artifacts for μC/OS, its internal systems and process are tuned to do so with the least possible expense.

VSC has developed certification packages that have enabled device certification for the dominant international safety-critical software standards used in aviation, medical, industrial, and building automation, and transportation. In addition, with advanced notice, VSC can rapidly satisfy many derivations used in some of the smaller niche markets.

In addition to expertise in RTOS kernels, memory management units, communication and file system software verification and validation, VSC provides a broad range of professional services including BSP and firmware development, run-time library validation and compiler qualification.

VSC also trains and mentors some of the largest companies in the world on certification and compliance strategies.

VSC guarantees its products; any software components that it supports are 100% compliant with relevant standards and WILL receive certification. VSC has never had a project fail, or a project canceled due to use of its products. Furthermore, VSC's COTS based pricing and licensing approach to safety-critical development consistently brings its offering to market at a fraction of the price of its competitors.

µC/OS Software and Certification Ready Evidence

VSC provides two tiers of certification products for Micriμm's µC/OS real-time operating system, Validation Suites™, and Validation Kits™. Validation Suites are the gold standard of certification support. From an ease of use perspective, a Validation Suite requires zero effort on the customer's part. Each Validation Suite is delivered running and tested on the evaluation hardware selected by the end customer. When it is time to apply for device certification, one simply references the Validation Suite in the master document set and includes a copy of the Validation Suite with their system submission.

The µC/OS kernel* has two parts, a "Core" component (~90%), and a "Port" component (~10%). The core software component provides the µC/OS kernel services and API's that are used by the application software. The core µC/OS software is static (hardware and compiler agnostic) and only changes due to planned updates and enhancements. The µC/OS port software component is the hardware abstraction layer that binds the core µC/OS software to the hardware. The µC/OS kernel is extremely portable and can be used with most 16 and 32-bit processors, DSPs, and FPGA softcores simply by adapting the µC/OS port software to the new hardware and toolset (compiler).

"µC/OS Core Software"	**"µC/OS Port Software"**
Approximately 90% of the µC/OS kernel software	Approximately 10% of the µC/OS kernel software
Contains all the µC/OS Kernel API's and services	Glue-layer between the hardware and the **"**µC/OS Kernel Core software
Abstracted away from hardware and tools	Dependent on hardware and tools
Does not change from target to target	Changing tools or hardware requires changes to port software

Similarly, the Validation Suite has both a "core" and a "port" component. Like the µC/OS kernel, approximately ninety percent of the core Validation Suite is static; it only changes in response to updates, and enhancements made to the core µC/OS software. The core of the Validation Suite represents over sixteen person-months of expert development effort. Like the µC/OS kernel core, it is agnostic to both the hardware and the toolset. Equally portable, Validation Suites can be "ported" to support µC/OS certification on almost any hardware and software.

To create a custom Validation Suite, the Validation Suite "port" is modified in support of the µC/OS port (port hardware). Reliability and accuracy is ensured by using same hardware target, toolset, µC/OS software version and the port software that is used in the end device. The results of the PST i.e. unit test results, integration test results and related test items are then collected and summarized. Project checklists, test reports, and all documentation are examined for correctness and documented with VSC signoff signatures. All supporting evidence is packaged

i.e., safety manual, software accomplishments summary, trace matrix, and all other relevant items required to complete the Validation Suite. In addition to interim deliveries that may have been made during the project, the final Validation Suite DVD is created and delivered via bonded postal service.

Validation Kits are for customers who desire to do final testing and gather their results in their own facility. Unlike the Validation Suites, Validation Kits do not provide the certification evidence based on the final hardware target. The customer "ports" the Validation Kit to their hardware, then executes their test code and gathers the results. This is the same exact process that VSC uses when they develop a Validation Suite. As shipped, each Validation Kits represents approximately 90% (~16 man-months) of the total µC/OS component certification effort.

VSC has Kits and Suites that support µC/OS-II (and the products below) certification in for the following markets and standards:

- RTCA DO-178C, DO-278A, EUROCAE ED-12C, ED-109A – Avionics (Airborne)
- IEC-61508 – Industrial control, automation, manufacturing, safety systems
- UL-1998 – Home, consumer, industrial safety
- IEC-62304 – Medical, clinical and home diagnostic and therapy
- FDA/CDRH 510(k) – Medical clinical and home diagnostic and therapy
- CENELEC EN-50128 – Rail/transportation, locomotive, signaling, and control

(for a full list of support please contact VSC at info@validatedsoftware.com)

When facing the decision to use in-house resources to develop the aforementioned products, consider the continuous investment made by both Micriµm and VSC in their product lines. The initial investment represents man-decades. Then consider the number of users and devices in which their products have been proven in over the last decade. Finally consider the wisdom of using precious resources to develop commodity infrastructure when they can be used to add the "special sauce" that makes your company successful today.

The world of the embedded engineer has become a much more stressful, and complicated place. Micriµm and Validated Software can show you how to reduce your risk and improve your time to market with their cost effective RTOS and certification products.

Visit Validated Software.com for the latest product information.

For more information about µC/OS-II, refer to www.Micriµm.com. For more information about the Validation Suite, see www.ValidatedSoftware.com. Complete product information, including sample of each DO-178C document in the Validation Suite can be requested from the Validated Software Sales office. Email Sales@ValidatedSoftware.com.