Inspection of Safety Critical Software: Expectations of the Regulator

G.J.Kurt Asmis PhD, P.Eng.[1]

5467 Riverside Drive
Manotick, Ottawa
CANADA, K4M 1H2
613 692-4656
kurt.asmis@magma.ca

Abstract

Chances are that if software is used within a device, system, vehicle, airplane or facility that can have an effect on safety, health, security or the environment regulators will be involved in the approval process before the entity can be released for public use. The regulators primary need is to have evidence & demonstrated assurance in place that the software meets requirements and does nothing unsafe. Regulators place heavy emphasis on testing, comprehensive & systematic inspection, standardized processes and qualified people

This paper presents a model of regulatory software approval that has evolved over the last 15 years: first, by the licensing of the safety critical shutdown systems software at Darlington which was completed in 1990; and second, by a period of formalization & standardization to bring software to the level of standard as other, albeit complex, engineered components. The thrust in the standardization process was to reach an acceptable level of maturity and remove fixation on software as a "treat-as-something-special" submission. This would permit software components to be changed, maintained or replaced as other engineered components are through an approved & managed process.

This paper goes behind the scene to show how regulators arrived at their "software" expectations. These expectations cascade down & influence the inspection efforts. The inspection of software is an important stepping stone in providing the evidence & confidence that the end result will be reasonable assurance of safe operation.

1. Introduction

Intuitively, I think, this audience feels inspection is an important component of the regulatory process. Regulators are inspectors and what could be more relevant then the inspection reports that this audience produces. I first explain the regulatory process that leads to the approval and use of equipment, systems or facilities of which the software is an important component. It is important to remember that regulators are not software engineers. Regulators respond positively to good evidence, and by extension to you, that what they are about to approve will cause no harm.

There are many regulatory agencies. My experience has been with regulators that deal in health, safety, security and protection of the environment. The model I draw from is the nuclear

[1] Dr. Asmis retired from the Federal Public Service of Canada, Canadian Nuclear Safety Commission, May, 2001. He was previously the Director of the Safety Evaluation Division (Engineering) which was responsible for the regulatory acceptance of safety critical software.

regulatory model. But drafting agencies that produce the laws and regulations for the nuclear regulatory agencies are similar to those of other safety agencies.

Regulators stake out their areas of responsibilities by excluding the public from certain activities or possession of certain materials. For example, in nuclear regulation it is forbidden that the general public possesses and manipulates radioactive materials without a license. This means that to it is not possible to operate a nuclear generating station and by extension the equipment that includes the software, that you might have inspected, without having gone through an approval process.

A regulator is not a design authority. Nor is a regulator an operator. It is the licensee that is responsible for safe design and operation. For example, a pilot has the primary responsibility of the safety of the airplane that he or she flies. It is not the regulator, Transport Canada that will be held responsible for mishaps or accidents. The regulator is another chain in the safety link. It is another level of protection that the public has decided it needs for its own safeguard.

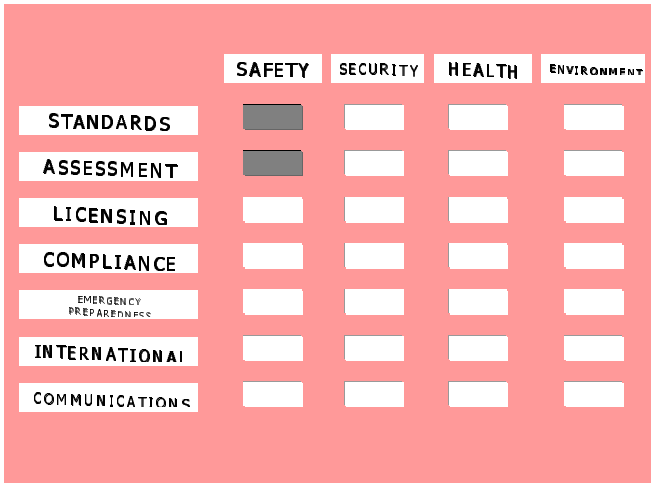
A regulator fulfills his or her mandate by asking for evidence that all is well. What constitutes proper and sufficient evidence is usually given to the discretion of the regulator. The dilemma is that for new technology or a new branch of engineering such as software engineering consensus standards do not exist. The regulator then, in search of evidence, may have to step in and be much more proactive in what is required than in any other mature technologies.

In addition to granting an operating license the regulator is also charged with responsibility of seeing that the conditions of the license are being met throughout operation. This brings into action additional inspection activities for the maintenance, addition or replacement of software throughout the life of the licensed activity.

2. What Do Regulators Do?

Regulators are busy people! The regulation of an industry where high reliability is demanded is complex. Not only does the equipment have to meet certain standards but there will be requirements on the licensing of operators, of maintenance, on financial guarantees, on the protection of the environment and on the need to communicate with the public. Technical requirements are as diverse as fire protection, the structural integrity of pipes, security and screening of workers.

The four by seven matrix on the right shows a typical regulatory framework. What the regulator has to do -- the mandate -- is written across the horizontal axis. How the regulator goes about his or her work is shown in the vertical axis. The flow of work leading to license approval starts with the specification of standards or the agreement of standards that will be used for the project. The licensee then carries out the design which is submitted for assessment. A license is then issued, most likely with conditions,



and the facility is put into action. Once the equipment or facilities is in operation the regulator will inspect as required and if necessary issue enforcement action. Other functions of the regulator are emergency preparedness just in case the previous four processes fail, the fulfillment of international obligations and the building of trust with the public through communication.

From the software perspective the first two boxes in the upper left-hand corner are the important ones. If the standards are in place, then the assessment process for the design will be fairly straightforward. Components and systems are compared against standards and if a match can be made the licensing process can proceed. When standards are not in place, or the standards or the effectiveness of the standards have been put into question because of incidents and accidents , then all of the vertical boxes included in the safety column become important. And particularly important will be the lower left box regarding public communications and public trust. During the licensing of the Darlington nuclear generating station the public eye was very much on the software component of that station. A heavy burden was placed on all participants to provide the evidence which would be defensible and stand public scrutiny.

Regulators abhor ad-hoc processes. They continuously strive towards formalization and standardization. This way, they can handle the enormous job entrusted to them by the public!

Regulators triage their time and effort according to risk! The regulator will take action and will adjust the intensity of his or her action depending on:

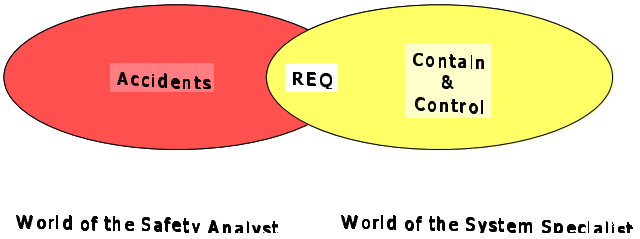
- will failure cause immediate and serious harm to the operator or the public
- will failure cause long-term negative effects
- will failure cause economic hardship or deny needed service to the public
- will failure cause loss of public confidence in the regulator and in the government

Regulators are scenario thinkers! And when they think up a scenario they usually are pessimists. The classical scenario in nuclear safety is that of pipe break. Loss of structural integrity of the primary heat transfer system is the dominant thinking of regulators. The pipe break causes loss of coolant, loss of heat removal to the fuel and the potential of uncontrolled dispersion of radioactive materials. We know from real accidents that things almost never happen according to the design scenarios but we also know that the defenses built-in to cater to the design scenarios usually do well in real events.

3 **{ scenario, liklehood,consequence}**

Regulators think out a scenario, work out the consequences, and if the consequences are deemed to be serious they determine the likelihood of that scenario taking place. The scenario thinking may be supported by mathematical tools such as event trees and fault trees, engineering judgment based on past experience, by intuition or by fundamental review of the physical phenomena that could take place. The regulator builds up his or her risk picture through the summation of the scenarios. With this triplet (scenario, likelihood, consequences) the regulator forges strategy to set standards and to carry out assessment and inspection.

Regulators expect the components of safety critical systems are rigorously inspected! And they expect the inspection (analysis) to verify that the specified behavior is met by the design.

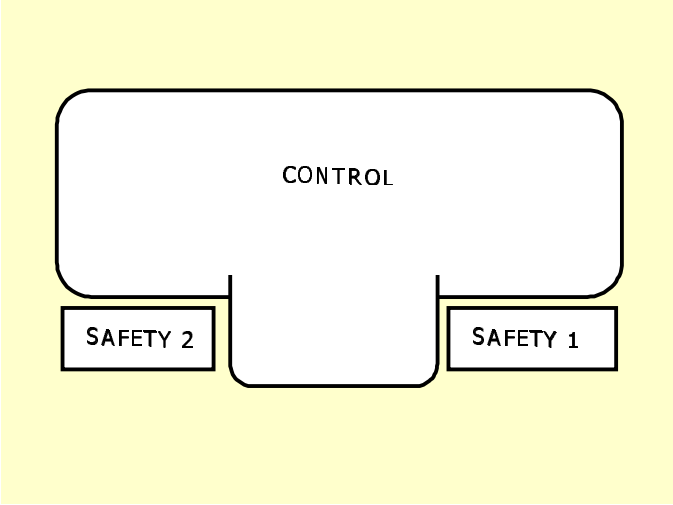


Regulators are used to having the hypothetical accidents -- and the resulting challenges and forces acting on a plant -- described in mathematical terms. For example, in accident analysis, an accident scenario will consist of a hypothetical pipe break within containment that will cause an overpressure condition, rise of temperature, rise of humidity, the injection of contaminants into containment and a host of other system transients. The safety analysts will describe these physical phenomena by mathematical equations . The regulator will expect that the corresponding containment and control actions that will render the plant safe against accidents, are also described in mathematical terms. They then expect that the proponents present conclusions that the facility is safe to by showing that the facility's withstand capabilities equal or exceed the anticipated challenges. Instrumentations and control logic required to button up containment will be shown to be adequate by logic equations and functional relationships. Demonstration of structural integrity to resist accident forces will be by mathematical analysis. It will not be adequate to simply inspect the design by reading blueprints, provide arguments of quality or appeal to previous successful designs.

$$C = \mathrm{REQ}(M)$$

The safety analyst will turn over to the system specialist the set of environmental variables of interest that need to be monitored and controlled. The transmission of requirements be in the form of a mathematical specification called REQ. The regulator upon concurrences will expect the safety systems to perform accordingly. The job of the operator, the job of any automated control system, is to monitor the accident variables deemed important and so identified by the safety analysts and to initiate the requisite control actions..

Regulators are not designers - but they have a mindset! In the design of high reliability systems regulators expect to see redundancy to cope with equipment failures and expect to see redundancy and diversity where exceptionally high reliability is required. They also expect to see excess capacity and extra strength to cope with the unforeseen.



When it comes to complexity, regulators like to divide and conquer. A case in point is the Canadian system of "special safety systems". There's a lot of complexity in the control of a nuclear generating station. There are many potential failure modes. They cannot all be guarded against, but protection can be provided by fairly simple systems whose only job is to contain or to safely shut down the reactor. Shutdown of the reactor can be controlled by three systems. The main reactor regulating system is responsible for adjusting power produced as well as carrying out other duties such as monitoring, alarming, logging and ancillary control. This system monitors the plant's performance and can cause power generation to stop if anything goes wrong. In addition there are two safety systems: there only job is to shut the reactor down if anything abnormal occurs. During normal operation the two safety systems have no function other than collecting and displaying data to be used in determining if they are functioning normally.

3. What are the "Software" Concerns of the Regulator?

Regulator will have heard of the Therac 25 cancer patient incidents. They will have heard of problems with fly-by-wire airplanes in particular the Airbus A320. The regulator may have heard of the fueling machine incident at Bruce A or of the Ariane rocket accident. The regulator will be concerned that software is close-coupled into potential accident chains.

When the Atomic Energy Control Board was required to make a decision on the safety of software at Darlington it already had available a list of standard scenarios and their consequences. Many of the scenarios required the shutdown systems to respond properly in order to limit the consequences to acceptable values. What we didn't know is what were the failure modes of software and their likelihood. In particular we needed to know if the normal defensive strategies of high reliability engineering such as redundancy, diverse design, design reviews, analysis and inspection, quality assurance and testing were effective in reducing the probability of failures. We also didn't know how to measure the reliability of software products.

What we discovered was that software produced with good engineering practices would certainly be better than one without. But we discovered that software had certain inherent characteristics that required measures to be tailored to those characteristics. For example, it would not be appropriate to blindly take the engineering practices shaped by the analog engineering world to the digital world.

Properties of software:

1) *Software is a "zero" tolerance product!* Software is not a forgiving product: the concept of tolerance does not exist! In an analog world there is a tolerance that allows for accepted behavior within parameters. A slight deviation in a property such as thickness, chemical composition would not o be a problem. The engineer designing the system depends on this. In software an apparently slight deviation such as a ";" is replaced by a "," a million line program can create unpredictable and unwanted responses.

2) In an analog world there are number of principles that in effect say "close is close enough". But, software does not exhibit continuity. This means that if the behavior of software is measured by two test points that appear to be very close to each other there is no guarantee that any intermediate points that have not been tested will give the expected response. In nuclear safety when we require the licensee to do a proof pressure test of containment of 1.15x design load we expect that the structure will "work " for all pressures up to the level and even for a fair distance beyond. In software testing is purely anecdotal.

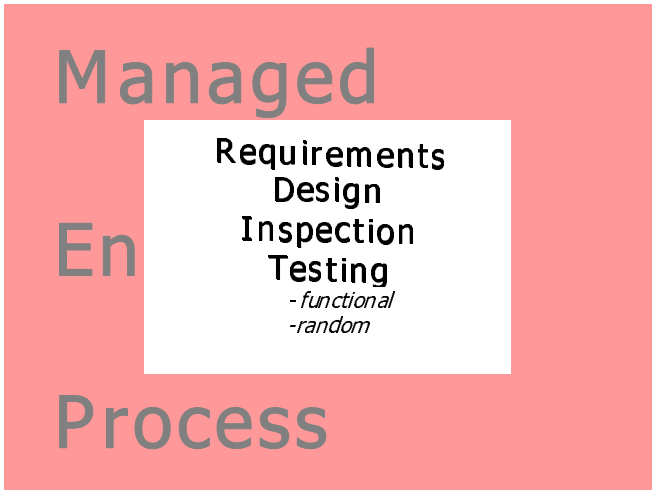
3) With software, water can flow "uphill"! Software does not have to adhere to physical laws. A software simulation of an analog mechanism does not need to obey any physical laws. This means that there are no natural constraints which help the designer.

When it comes to demanding high reliability, regulators look for the use of redundancy, diversity and excess capacity. Applied to software, redundancy is a non-starter as repeats of the same software will simply generate the same error. Diversity, the construction of parallel software systems with different hardware and different software development, appears to be a viable alternative but close examination has shown that software developers developing the products independently correlate their errors!

Excess capacity or overrating certain components to provide high reliability has its parallel in software by providing extra error checks, redundant code and error correcting strategies which while they might solve some problems will cause a more complex product to be created and increase the analysis and testing burden.

Given the unique characteristics of software we created our approval strategy. That strategy realized that software is a technology on the rise and had certain requisite properties that had safety advantages that we were not prepared to forego. The strategy recognized that the decline of analog control technology also had negative safety consequences. The demands of high reliability and the specific characteristics of software demanded that we adapt our expectations accordingly to include this new technology. One of our key strategies was to place "inspection" on a rigorous, mathematical footing.

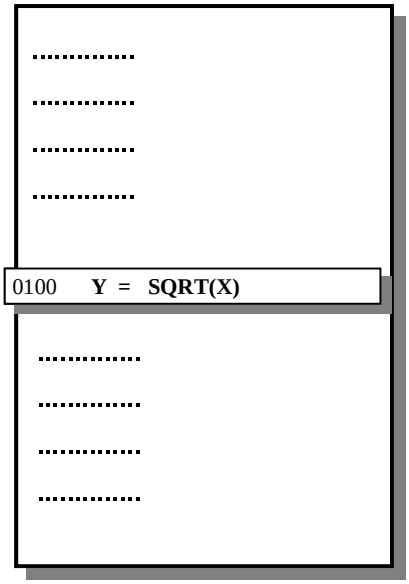
4. Expectations for the Inspection of Safety Critical Software



The regulator needs to see evidence on which to base decision. The primary evidence is documentation, but evidence could include witnessing of test results, and the knowledge that other regulators have accepted the same product.

For safety critical components, systems and structures the regulator wants to be assured that a recognized engineering process has been followed and that that process has checks and balances along the process trail that give reasonable assurance that all safety requirements have been included into the finished product. The managed engineering process would include standards that guide the production of documents, standards to provide for training of staff, standards that will be used to guide testing and commissioning. As well, standards will need to be included for harsh environment qualification and human/machine interface design based on human factors principles. In short, all that is necessary for a quality, safe product!

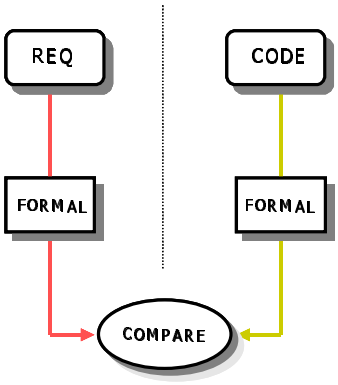
Regulators expect inspections to be mathematically based! In software the devil is in the details! The inspection techniques must be accurate enough to examine the fine details of programming as well as powerful enough to capture functional behavior. The licensee needs to build up a safety case: this means that the inspection process must have the capability to examine the fine detail of assembly language as well as compactly describe the behaviour of large " chunks " of code. The aim is always to verify correspondence with safety requirements. The inspection process has to confirm that the software will carry out its required safety functions and not do unintended, unsafe actions.



Mathematically based inspections are possible because a sequence of program statements defines a mathematical function. This fact can be most easily understood by starting with a single executable line. In the example: y = SQRT(x). With certain preconditions, such a single statement can be regarded as a function that transforms an initial data state into a final data state. The concept of a function can be expanded from single statements to sequences of statements, procedures or subroutines. It is therefore possible to talk of the program function that defines a state transition from initial data states to final data states provided that the execution terminates. These program functions are displayed in table format.

The program function is a representation of the outputs of a sequence of program text. Implementation preferred details as well as the values or expressions of local variables created to assist the calculational process has been removed. The

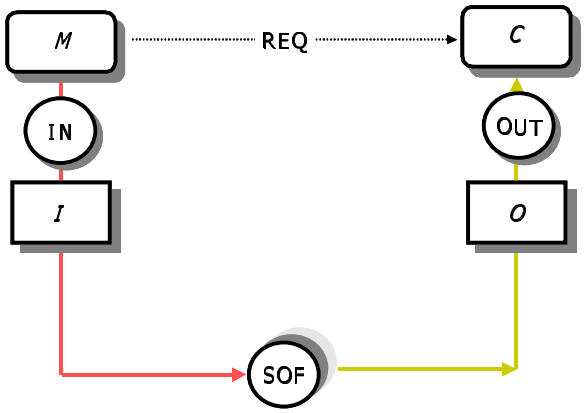
program function improves the inspection of code by focusing on the outcome. The usual, but error prone, methods of code inspection by line by line reading is not needed except to provide a measure of code quality.



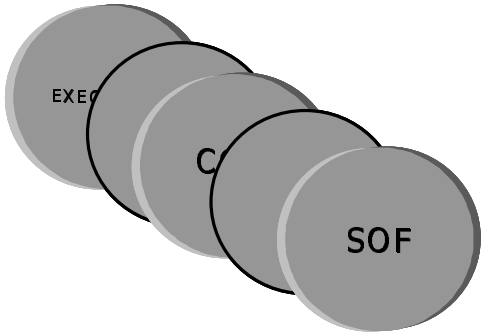
The actions predicted by the program function can be compared to the actions required. In the original Darlington inspection process systematic mathematical manipulation of both program function tables and the requirements were carried out. The inspection process continued until the action of output variables were shown to be identical to the actions specified within the allowable tolerances.

In the original Darlington approval process the entire program text of both shutdown systems was reverse engineered and formaly compared to the requirements document. The process was manually intensive and exacting.

Regulators expect a "forward" software development process! Following the approval of the original Darlington software, a period of formalization and standardization took place. The document structure that evolved and that gained consensus by the regulator, the AECB (now the CNSC), by the designer, AECL, and by the utility, Ontario Power Generation is shown here.



The new process is a forward software development process that creates a class of documents specifically tailored to the unique requirements of software in computerized systems. The documents IN and OUT describe the behaviour of the input and output devices. The document SOF describes the behaviour of the software that achieves what is required by the document REQ.



The four major documents describe "what" is required. Each class of documents can generate a series of work products in ever increasing specificity. For example, "underneath" the document **SOF** are a series of work products. The dominant characteristic of a work product is that it begins with requirements from the previous stage, transforms that requirement into an output which is then used for the beginning of the next stage. The process stops when a product is built or purchased which has thea requisite functionality without unwanted side effects.

This layering process allows an informed, risk based inspection strategy to be devised. Inspections, and the intensity of the inspection can be adjusted to the reliability of the tools that transform input to output. For example, in the original Darlington licensing action, the code was handwritten based upon informal natural language specifications. This is where we felt that rigorous inspection was needed. Other tools were used to transform program text into executable code. These tools, we felt, had a high degree of reliability - demonstrated in service - and detailed inspection was not warranted. These tools we checked through the testing process.

Regulators expect continuous improvement! Although regulators are by their very nature conservative, regulators do not like to be caught short! Software approval is an intensive and expensive process. There is much software in a nuclear power generating station that could use the benefits of well thought out inspections. Not all inspections need to be formal, mathematical inspections. For much software there will be an overall safety advantage by increasing the usability, availability and reliability of that application. For example, the reactor regulating system, that controls the reactor, has not been formerly inspected. The techniques applicable to safety critical software are simply too intensive for this kind of application. Nevertheless, the overall reliability of the station could be improved if related techniques are used more widely. Increasing the reliability of software throughout the station will be of benefit if the overall need for the special safety systems is decreased.

Graphical languages should be investigated to see if they can decrease the level of inspection required. Graphical languages have the promise of combining requirements and code.

Accident records show that the human element at the level of operator and maintainer interface is important. Inspection techniques should be extended into this arena: getting the software correct and then using this software correctly are complementary activities.

5. Conclusions

This paper presented a model of regulatory software approval that has evolved over the last 15 years: first, by the licensing of the safety critical shutdown systems software at Darlington which was completed in 1990; and second, by a period of formalization & standardization to bring software to the level of standard as other, albeit complex, engineered components. The thrust in the standardization process was to reach an acceptable level of maturity and remove fixation on software as a "treat-as-something-special" submission. This permitted software components to be changed, maintained or replaced as other engineered components are through an approved & managed process.

Regulators need to have evidence & demonstrated assurance in place that safety critical software meets requirements and does nothing unsafe. Regulators place heavy emphasis on testing, comprehensive & systematic inspection, standardized processes and qualified people. Inspection of software is an important stepping stone in providing the evidence & confidence that the end result will be reasonable assurance of safe operation.

Acknowledgments

I would like to thank Artur Faya for valuable discussions that input into this paper. Dr. Faya and his team were the prime mover within the Commission for the Darlington shutdown systems trip computers software redesign. Thanks to Dave Parnas to take the regulators expectations and show by examples related to our experience in conventional engineering that software was amenable to mathematical analysis with tools that we already knew. It was the staff of Ontario Power Generation, then Ontario Hydro, and AECL that took these concepts and turned them into practical, viable engineering tools.

To read further

C. Chun, L. Staples and A.J. Faya, Regulatory assessment of the Darlington shutdown systems trip computer software redesign, International topical meeting on nuclear power plant instrumentation, control and human-machine interface technologies, Washington DC, November, 2000.

IAEA, Safety Standards Series, Software for computer-based systems important to safety in nuclear power plants, Safety guide, No. NS-G-1.1, 2000.

Canadian Nuclear Safety Commission, C--138, Draft regulatory guide, Software in protection and control systems, October 1999.

OECD, NEA, Committee on the Safety of Nuclear Installations, Operating and maintenance experience with computer-based systems in nuclear power plants, Report by the PWG-1 Task group on computer-based systems important to safety, 1998.

Health and Safety Executive, Four party regulatory consensus report on the safety case for computer-based systems in nuclear power plants, Canada, France, United Kingdom, United States of America, November 1997.

D.L. Parnas, G.J.K. Asmis, J.Madey, Assessment of safety critical software in nuclear power plants, Nuclear safety, Volume 32, No. 2, April – June, pp 189-198, 1991.