Life Is On | Schneider Electric

Automated validation and testing of application logic in safety logic solvers

White Paper

by Steve J Elliott, Senior Marketing Director

Executive summary

SIS systems all require software logic testing and periodic re-validation. Manual testing can be laborious, time consuming and error prone and may not always be effective in finding certain classes of defect. Automated testing and re-validation offers a possibility to perform these types of test more efficiently. While, automated software testing has long been accepted as a best practice when developing and testing commercial software code. That same principal can now be applied when testing and validating the application logic in safety instrumented systems.

Introduction

"In today's economic climate everyone strives for productivity, efficiency and quality gains. The use of automated testing can be more efficient and effective than traditional manual validation efforts"

When implementing any safety instrumented system, rigorous testing and documentation of the application logic is a critical part of realizing any safety instrumented system.

In today's economic climate, everyone strives for productivity, efficiency and quality gains. The use of modern automated tools and techniques achieve faster time to quality, significantly increase the likelihood of a trouble-free factory acceptance test (FAT) and successful start-up, as well as reduce overall project life cycle cost.

With approval of international standards such as OSHA PSM 1910, ANSI/ISA.S84.01, IEC61508 and IEC61511, testing has become mandatory to maintain a pre-defined Safety Integrity Level (SIL). Today, tools and processes are available to guarantee consistent, complete and documented testing of the logic, in less time than ever before.

Additional benefits can be obtained throughout the operating life of the safety instrumented system when the same automated tools and techniques can be used for periodic proof testing of the logic solver, operator training, validation of changes and modifications, and assistance with any SIS related alarm rationalization.

This white paper outlines some of the areas for consideration when deciding to use automated test tools and techniques, including:

- What the safety standards say
- What to consider when selecting a test tool
- Defining clear objectives and developing a strategy
- When to use manual versus automated testing
- How to use automated testing to create additional value
- Considerations for governance and controls
- Understanding potential risks
- Some of the pros and cons of automated testing

What do the standards say?

Any standard, either local or international, put simply is good engineering practice and provides a systematic and consistent framework. With the acceptance of international standards such as ANSI/ISA.S84.01, IEC61511 and IEC61508 verification, validation and testing have become mandatory during the execution of the safety lifecycle.

Using one of the most commonly adopted standards IEC61511: Edition 2 as an example greater emphasis has been placed on application software (Clause 12 provides a section dedicated to SIS application program development).

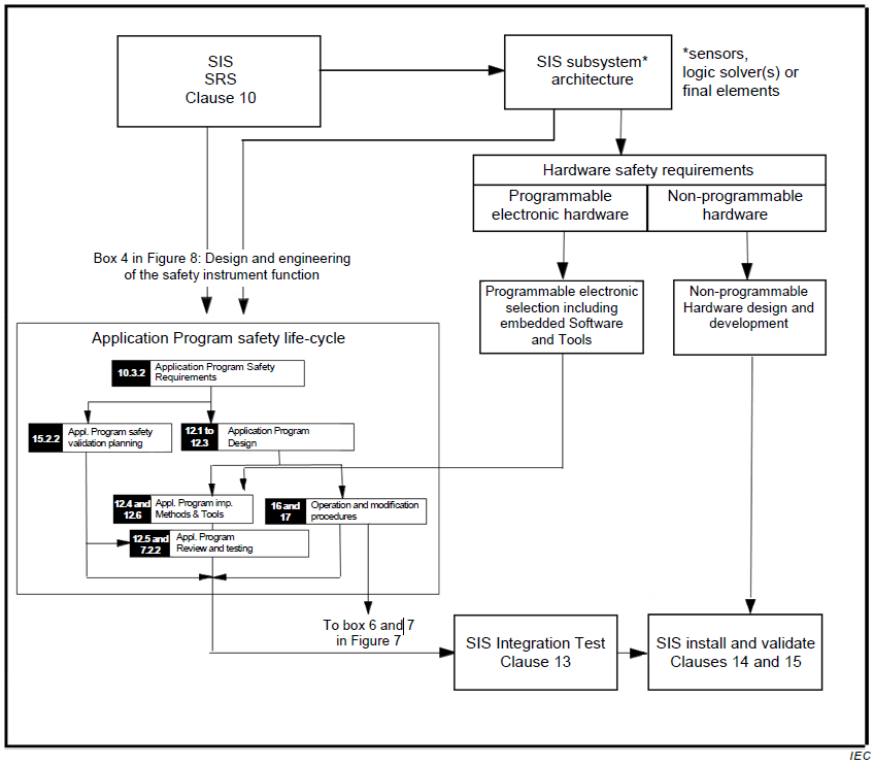

Figure 1

IEC61511 Edition 2, section 12 Application program safety lifecycle and its relationship to the SIS safety lifecycle.

In particular IEC61511 2 edition, section 12.5 requirements for application program verification (review and testing) states the need to address “testing for failure conditions (i.e. negative testing)”; re-verification on requirement changes; re-testing upon test failures.

This can be complex, time consuming (and sometimes not realistically possible) when testing detector voting schemes e.g. 2*oo*16 (two out of sixteen) thermocouple trips on a hydrocracker reactor bed where there are almost infinite combinations to be tested for both correct operation as well as negative operation.

Using traditional manual methods are not practical to check all combinations; so spot checks are typically performed. Using automated methods and techniques addresses this requirement and allows users to test every combination thoroughly.

In addition to Clause 12, Clauses 14, 15 and 16 require that the SIS is commissioned, operated and maintained in a way that sustains the required safety integrity. This will require periodic revalidation of the commissioned SIF’s (which includes the SIS logic solver and application code). Scripts developed during the initial testing phase can easily be modified to be used for any re-validation requirements of the logic.

What should I consider from a test tool?

Automated test tools are constantly evolving. However there are some fundamental basics that should be considered when looking for any test automation tool.

Useable across the Functional Safety Lifecycle:
The testing framework should offer the capability of not only being used during application testing as defined in Clause 12 but also during periodic re-validation of commissioned systems as defined in Clause 14 and 15.

Simple interface for creating automated tests:
The testing framework should offer a 'point-and-click' interface for creating, executing and interacting with the application components under test. Testers should be able to easily and intuitively create and edit tests, reuse proven test bocks where possible, be able to visualize each step i.e. clearly see what is being tested and see what the results are.

Ensure scalability:
A key benefit of automating functional testing is the ability to test large systems and data points. Make sure that you are able to manipulate the test sets and quickly create hundreds of test iterations and permutations with minimal effort.

Concise reporting:
The ability to run high volume of tests is of little benefit if the results of the test are not clear and easy to understand. The toolset should automatically generate reports of the tests run and show the results in an easy-to-read format. The reports should provide specifics about where tests failed and what test data was used.

Coupling tests to the application under test:
A key factor to consider when considering a test tool is to identify if it is to be tightly integrated or loosely coupled to the application under test:

- Tightly integrated systems are normally designed, developed and tested by the manufacturer for specific use with their systems. For example, Triconex Safety Validator is design to work with Triconex Report Generator, Triconex TriStation TS1131 programming software, the Triconex Emulator and the actual Triconex family of logic solvers. Benefits can be realized such as seamless data exchange between applications improving time, efficiency and reducing error.
- Loosely coupled systems are often designed by third party vendors for general use with any manufacturers equipment. As such they may provide more flexibility, but are rarely certified (the emphasis is placed on the user to ensure correct implementation) and need greater care and attention to understand exactly what functions can (and more importantly) can't be done. They often require more manual configuration effort which introduces the likelihood of human error.

Create additional value:
Consider how else the test cool can be used to create additional value. For example, the ability to interface with third party HMI's to test proper operation of the graphics, alarm prioritization and common database as well as the capability of creating a low resolution operator training simulator used in conjunction with the HMI.

Classification and requirements:
When using any software tools as part of the safety lifecycle it is important to understand their role and influence on the safety instrumented system, and the potential impact or consequence that they may.

When assessing the requirement for any tool, it is important to determine the level of reliance placed on the tools, potential failure mechanisms of the tools that may affect the executable software, and the appropriate mitigation measures, if applicable.

IEC61508-2:2010, Part 4 – Definitions and Abbreviations has a very clear classification of software tools used in safety system applications. Tools are classified as either online support tools (can directly influence the safety-related system during its runtime) or offline support tools (supports a phase of the software development lifecycle and cannot directly influence the safety-related system during its run time).

3.2.10
software on-line support tool
software tool that can directly influence the safety-related system during its run time

3.2.11
software off-line support tool
software tool that supports a phase of the software development lifecycle and that cannot directly influence the safety-related system during its run time. Software off-line tools may be divided into the following classes:

– **T1**
generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system;

NOTE 1 T1 examples include: a text editor or a requirements or design support tool with no automatic code generation capabilities; configuration control tools.

– **T2**
supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software;

NOTE 2 T2 examples include: a test harness generator; a test coverage measurement tool; a static analysis tool.

– **T3**
generates outputs which can directly or indirectly contribute to the executable code of the safety related system.

NOTE 3 T3 examples include: an optimising compiler where the relationship between the source code program and the generated object code is not obvious; a compiler that incorporates an executable run-time package into the executable code.

Figure 2

IEC61511 Edition 2, section 3, Terms and definitions

For offline support tools such as automated test applications, there are further levels of guidance for sub-levels T1, T2 and T3. These indicate the behavior of the tool, instructions or constraints on its described use.

When dealing with safety applications and selecting a test tool, make sure that the tool has been developed in accordance with international standards, or even better, comes complete with independent certification from a recognized certification body such as TÜV.

Table 1

Tool classification and requirements

Software type (IEC61508)	Manufacturer View	System Integrator / Operator / Owner View
Online software: Software the runs to realize the function	The products that are developed	**Firmware** that runs in components **Application** software (including libraries)
T3 offline support tool: Software to generate the online software	Compilers that are used	**Tools** provided by manufacturers of PLC / Logic Solver
T2 offline support tool: Software for testing	Unit testing, metrics etc. (embedded)	Application Logic Test tools
T1 offline support tool: Office tools	Requirements engineering, office…..	Requirements engineering, office…..

Define your objectives before you start

Before undertaking any automated testing it is important to define the objectives of automated testing before selecting an automated toolset. This will help you chose the rights tools and the right foundation for your immediate and future needs. For example:

- What is the expected life of the tools?
  - One off for a specific project?
  - To be used for re-validation during the operating life of the safety system?
  - To be used for operator training
  - To be used for alarm rationalization assistance
- What percentage of the application will be automatically tested?
- Should the test application be tightly integrated or loosely coupled?
- What are the SIS logic re-validation requirements during critical manpower times such as during turn arounds?
- What are the company's requirements for re-testing logic after a SIS logic change on a commissioned system?
- What are the skills of the test team?
- Who will create and maintain the test scripts?
- How much is the organization willing to invest in automated testing?
- What is the expected return on investment (ROI) from the automated test efforts over the life of the safety instrumented system?

Discussions of these objectives will govern the required effort and ROI. Common considerations for automated testing include:

- Maximizing test coverage
- Scalable for future requirements
- Reliability and consistency of results
- Ease of maintenance
- Duration of execution
- Repeatability of tests
- Reusability of test scripts
- Around the clock re-validation during turnaround downtimes
- Re-test/validation requirements after a SIS logic change
- End to end testing throughout the safety system lifecycle

Develop a strategy

Developing a test automation strategy is very important in mapping out what's to be automated, how it's going to be done, how the scripts will be maintained and what the expected costs and benefits will be.

The effort of test automation is an investment – on any safety instrumented system project, a detailed test plan / factory acceptance test plan will be developed.

To understand automated testing one must understand traditional manual testing methodologies currently being used. Traditionally these testing documents are created in word or excel formats and detail each of the tests and steps of the tests to be performed.

For example:

Subject: XXX Factory Acceptance Test		**No.** FAT-XXX-XX
Rev. No. 0	**Date:** XXX	**Page:** 6-6

6.6 BURNER PERMISSIVES

The burner permissives (Table 6-6) should be checked by satisfying all conditions but the one being tested and verifying that the main burner cannot be started. Once all permissives are satisfied, "OK to Start Burner" will be displayed on the Burner Control MMI screen.

Burner Permissives	Perm OK State	Verify (√)
No burner shutdowns exits (Section 6.8)	BURNER-SHUTDOWN = ON	
Ignitor on for at least 30 seconds.	"Ignitor On" = ON > 30 sec.	
Burner Pressure NOT low and transmitter NOT failed.	MZP733 > 1.0 psig MZP733-FLT = OFF	
Air flow set for minimum fire	MZF204 = ON	
Fuel gas control valve in the low fire position	MZF206 = ON	

Table 6-6

Figure 3

Typical extract from a factory acceptance test plan

As these test plans have to be created as part of the original SIS project, instead of creating them in word manually, create them directly in an automated test automation tool. Even if the immediate payoff appears minimal, it provides the following benefits:

1. Because the test is automated, it eliminates the human factors such as testing sequence errors and fatigue that generally occurs as testing progresses.
2. Complete re-running of the test scripts any time a logic change occurs (no matter how minor).
3. The use from running these automated tests every time a re-test is required, or when periodic re-validation the safety instrumented systems, during turn arounds etc.
4. The scripts can be modified to inject errors in the logic sequence that can be used for operator training (when used in conjunction with a HMI).

Therefore, ensuring that the scripts can be easily maintained becomes very important.

Develop a written standard

A key part of the strategy when considering the use of automated test tools is to develop a written standard for testing scripts. The standard includes such topics as:

- Developing a testing flow standard to includes the number and types of test cases.
- Documentation and commenting requirements for each test case
- Uses and types of tiebacks within the test cases
- Understanding emulated vs. connected hardware testing requirements

Start small and build confidence

Sounds obvious, but when learning to use any new tool or applications it's common to make mistakes. One way to mitigate these mistakes is to start small and create test scripts that will provide immediate pay back. That is, create scripts which won't take too much time to create yet will obviously save manual testing effort.

For example, consider how to automate high volume, low complexity devices such as the field instruments (scaling, trip limits, under range, over range, fault, alarm, inhibit etc.) These scripts will be immediately useful. Those creating the scripts will learn more about the tool's functionality and learn to design even better scripts.

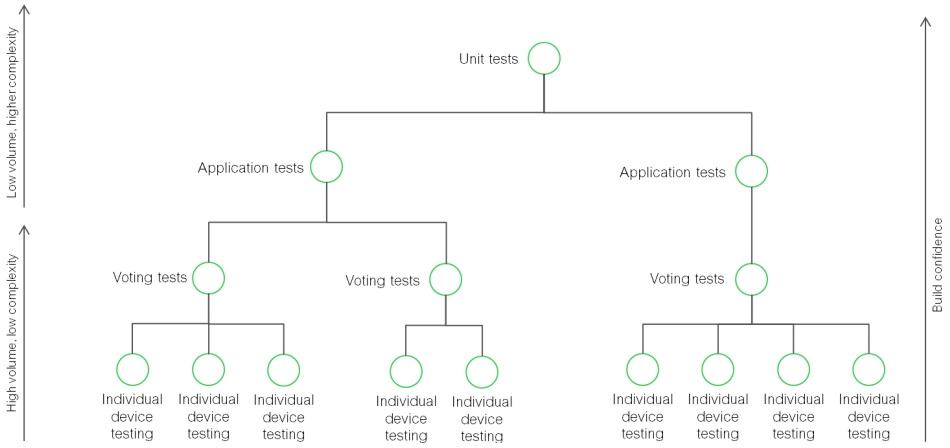

Figure 4

Start small at the high volume, low complexity level and build your way up

As experience is gained with the testing tool, a long-term approach to test automation can start to be developed.

As confidence builds, start to build up the complexity of testing. Again, start off small when designing scripts. Identify the functional areas within the application being tested. Design at a higher level how each of these functional areas would be automated, then create a specific automated test design for one of the functional areas.

If there are opportunities to use common scripting techniques with other testing modules, then identify these common approaches as potential standards would be useful in creating maintainable scripts.

To automate or not?

It is probably impossible to automate all aspects of a test. Automated testing compliments manual testing methods and should be focused on automating repetitive, low level tasks to free time to focus attention and resources on the high risk areas or where more attention needs to be focused.

Note: automated testing does not replace good test planning, writing of test cases or some of the manual testing effort.

It sounds obvious, but automate tests where it makes sense. Some software testing tasks, such as extensive low-level interface or device testing, can be laborious and time-consuming to do manually. In addition, a manual approach might not always be effective in finding certain classes of defects.

For example, when testing multiple detector voting arrangements such as Fire and Gas systems with X*oo*N (X out of N) detector voting schemes, or on a hydrocracking unit where there are often safety instrumented functions with 2*oo*12 (two out of twelve) or even 2*oo*16 (two out of sixteen) thermocouple trips with trip delay times on the reactor bed temperatures.

In these examples there are so many potential combinations of detectors to be tested that when testing manually, only representative sample is tested.

Test automation offers a possibility to perform these types of testing effectively. Once automated tests have been developed, they can be run quickly and repeatedly. Test logic where inputs are transformed to outputs without human interaction.

It is worth noting that just because a test can be automated, it doesn't necessarily mean that it should be automated. Even when there is clear evidence that automating test effort is economically justified, there may be valid reasons for manual testing. For example, you may want to use the testing to develop the skills, competencies of the team, expand the knowledge and understanding of the application under test or use as a means of training the team.

A good question to ask yourself is:
"If I automate this test, what will I gain? What will I lose? What will be the consequence?"

When should I test?

The relationship between incremental cost to the project as a result of finding application errors increases significantly depending on when they uncovered.

As a rule, you should test as soon as you can and as often you can! Lifecycle costs increase the later you test and the cost to resolve any issues increases exponentially.

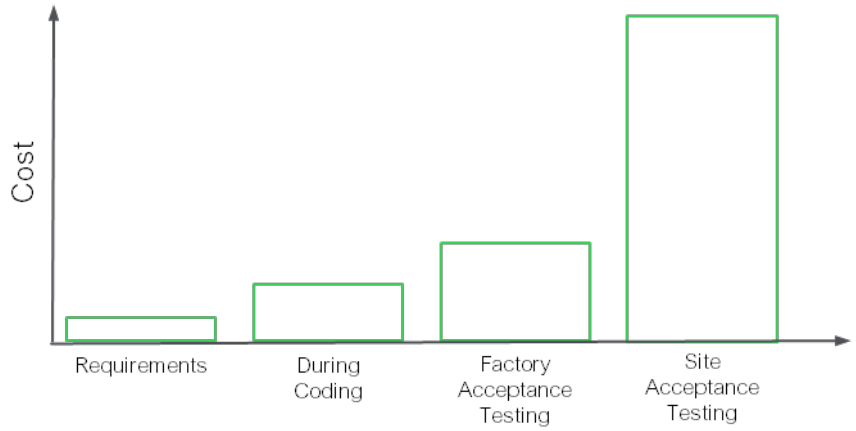

Figure 5

The cost implication of finding application logic errors and making corrections changes significantly across the lifecycle

Where ever possible, a phased approach to the test and validation of the application logic has many benefits. Although some time and money may have to be invested earlier in the project the return on these investments can be significant.

Cost reduction

The cost to develop the test plan, test scripts and the time to test each phase may appear to add to the overall cost. However, this would only apply if you could guarantee that the application logic was 100% correct first time i.e. no mistakes were made and no corrective actions were required between the initial design and the final installed application.

Schedule reduction

The impact of a change or corrective action early in the project can have schedule benefits. This seems to be common sense, but often compromises or shortcuts are made to get the physical equipment and application to the field as soon as possible. The most common schedule compromise is in proper and complete testing, when the project end date doesn't change and the scheduled testing time gets compressed.

For example, a simple 2*oo*3 (two out of three) analogue voting block may be required to meet the requirements defined in the safety requirement specification. This can become complicated when the requirements for bad transmitter faults, degeneration in voting, bypass requirements etc. are included. Testing this block early in the project means that it can then be confidently used where and when required. It may take some iteration to get the functionality correct, but this can be done in a controlled environment with a minimum number of people and no impact to anything else. The time to develop and test this blocks is minimal compared to the time it would take to detect and correct an error during start-up.

Quality improvement

A good way to look at the quality of the application logic development is to use the V-model as defined in IEC61508:2010 – Part 3 Software requirements:

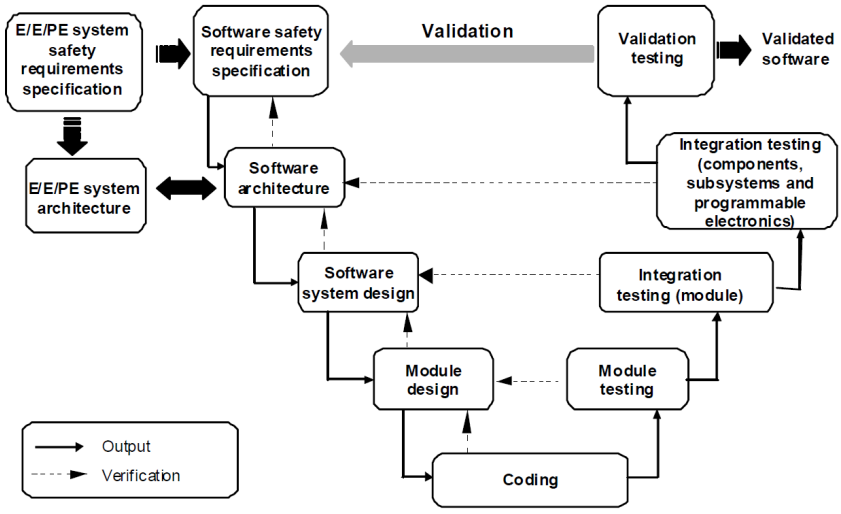

Figure 6

Software systematic capability and the development lifecycle (the V-model)

This provides a more structured and systematic approach for the validation process. It is broken down into smaller steps and allows for corrective action during the development process. When application logic blocks are developed, they can be tested or validated as a black box, without the need for the complete application. If an error is detected, corrective action can be taken early in the project. Another advantage is that sections that have been proven in other projects can be re-used without the need for another validation step.

Using this V-model approach, a structured test with a very high test coverage factor can be accomplished and the quantity of anomalies detected during internal testing, factory acceptance testing and start-up will be reduced. This approach also benefits future expansions and testing as the application logic is more maintainable.

Increased safety

Using a systematic approach as defined in IEC61508 and IEC61511 to application logic development and verification is one of the many ways to reduce the probability of unwanted events.

Create additional value

The use of automated testing can create additional value especially during system checkout. If the safety system is connected to distributed control systems (DCS) or a human machine interface (HMI) then the automated tests can be used for testing the operator interface, alarms, data historian etc.

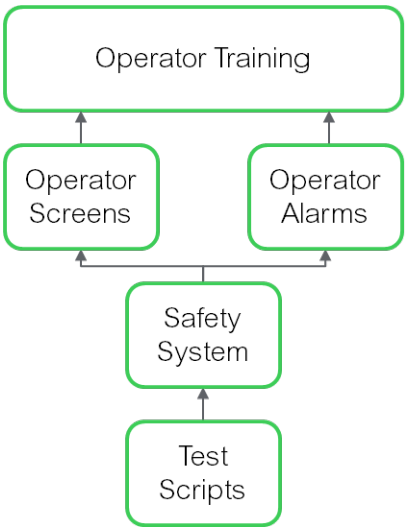

Figure 7

Use automated testing to create additional value to operations and maintenance

These same tests can also be used to aid the plant operators and gain valuable experience in a simulated environment. For example, test scenarios can be built for start-up, shutdown and abnormal operating conditions. Test scripts and scenarios can be used to test operating procedures. Operator performance can be measured on benchmarked including response times / reaction times under simulated abnormal operating conditions.

Test scripts can also be used for “what-if” scenarios e.g. what happens if this sensor fails, what happens if we remove this probe to support maintenance activities.

Apply Governance

Governance is a critical aspect of test automation design. Governance ensures that all areas of the testing methodology are kept consistent, regardless of the individual developing the test scripts, executing or documenting tests. Governance should include defining the process for development, testing, management and control of the test results and makes automated testing successful. Typical governance parameters include:

- Test suite structure
- Versioning
- Any applicable codes or standards
- Naming conventions

Table 2

Considerations when designing test scripts

Issue	Question	Consideration
Organization	How to organize the tests?	Group by safety function Group by modularity based on design of application program Group by tests that require human intervention
Sequences	Which sequences can be re-used?	e.g. call a reset sequence
Patterns	Which patterns can be re-used	e.g. 2oo3 voting logic can be used as a template
Execution	Which sequence should tests be executed in?	Group fully automated tests together Run over night
Lifecycle	What is the test coverage to overall safety lifecycle?	Include traceability back to the Safety Requirement Specification (SRS) as well as Process Hazard Analysis (PHA) to facilitate coverage and change impact analysis

Understand potential risks

Potential risks to automated testing are rarely associated to technology or the tools, but more to the human factors, the skills, competencies and behavior of the people using the tools.

Some common risks to the test automation effort include management and team members support fading if they don't see immediate results. Mitigate this by "starting small" and building confidence.

Demanding schedules will put pressure on the test team, project management and project funding to do what it takes to get the system tested and "out the door". The reality is that the next time you need to modify the system or re-validate the system (which usually has the same constraints) you'll wish you had the automated testing in place.

Another risk, certainly in the oil and gas industry, is the attrition of industry knowledge. It is estimated that nearly 50% of the workforce is within 5 years of retirement, and for every 2 people retiring there is only one new employee entering the workforce.

If contractors are used to help build the test automation effort because of their experience, there is the risk that much of the experience and skills will 'walk away' when the contractor leaves. If a contractor is used, ensure there is a plan to transfer the knowledge since this may affect the maintenance effort and new development of test scripts.

It is therefore critical to the sustainability of the safety systems that knowledge of the application is captured and retained for the operating life of the safety system. Capturing knowledge in the test scripts for future use is one such way of retaining knowledge and protecting the business in years to come.

Another potential risk is fatigue - users often find that human fatigue leads to errors and inaccurate results when conducting Factory Acceptance Tests (FAT) and Site Acceptance Tests (SAT). This problem is exacerbated when FAT durations last for several weeks and are under time pressure, so long working hours / shift patterns introduce fatigue which leads to short cuts, complacency and mistakes.

"Automated tests perform the same steps precisely every time, and never forget to document the test results."

Humans are good at noticing oddities, they are bad at performing repetitive tasks, painstaking or precise checking of results. Tools are good at catching what humans wont. Tools can analyze more than a person can see. The fact that humans can't be repeatable and precise means that repeated tests are often slightly different. For example, if a test fails and an error is found, can the test be re-run exactly the same each and every time? Automated tests perform the same steps precisely every time, and never forget to document the results.

Pros and cons of automated testing

To many people, the benefits of automation are pretty obvious. Tests can be run faster, they're consistent, and tests can be run over and over again with less overhead. As more automated tests are added to the test suite more tests can be run each time thereafter. Manual testing never goes away, but these efforts can now be focused on more rigorous test.

Advantages	Disadvantages
Save Time: Automated testing can be up to 50% faster than traditional manual testing methods Tests can be re-run in seconds or minutes, not hours or days.	
Optimize Resources: Automated tests can be run around the clock allowing you to test 100% of the day Multiple tests can be run in parallel Decouple logic testing from the physical hardware Testing can be performed anywhere in any location	
Ensure repeatable quality: Automated tests eradicate human factors and human error Consistent and systematic approach	**Management of change:** Governance and controls should be put in place to manage and maintain test scripts
Increase efficiency: Automated tests repeat the same steps precisely every time Subsets of tests can be run / re-run	
Increase accuracy: Automated tests deliver repeatable and consistent tests Results are automatically documented	Tests are only as good as the test scripts An independent peer review of the test scripts should be performed.
Increase integrity: Automated tests provide greater test coverage - test multiple combinations, scenarios, look for, negative outcomes Test modifications and changes before implementing	
Create additional value: Capture application knowledge and expertise in test scripts Use to test HMI screens and alarms Train Operators to deal with different scenario's such as start-up, shutdown, abnormal conditions Measure operator effectiveness, response times, reaction etc.	

Conclusion

With the right strategy, clearly defined objectives and governance in place the use of automated test tools presents an opportunity to improve quality, schedule and cost in the realization of safety instrumented systems.

The use of automated test tools and techniques should be seen as complimentary to manual methods – use the best of both worlds where and when it makes sense.

"Automating without good test design may result in a lot of activity, but little value."

An automation tool is important, but it's not the solution to everything. When strategizing for test automation, plan to achieve small successes and grow. It's better to incur a small investment and see what the effort really takes before going gung ho and trying to automate the whole application test. This also gives those doing the work the opportunity to try things, learn and design even better approaches.

About the author

Steve J Elliott is a Senior Marketing Director for Triconex by Schneider Electric's safety solutions. He is a TÜV certified functional safety engineer with over 20 years of experience in distributed control systems, SCADA systems, PLC and safety systems. He holds a Process Safety patent and has published multiple articles in global journals focused on functional and process safety and has authored several white papers.

Acknowledgements

Special thanks to **Phil Blanchard.** Phil is the Advanced Delivery Director for Schneider Electric. He is an authority in Functional Safety, and Alarm management with over 30 years of experience in the implementation of Safety Instrumented Systems and Safety Lifecycle.

Acknowledgements

Special thanks to **Ajay Mishra**. Ajay is the R&D Program Director for Triconex and helps define the detailed features and technology roadmaps for the Triconex safety and critical control products. Ajay is a TÜV certified Functional Safety Engineer for hardware/software design (IEC 61508) and Safety Instrumented Systems (IEC 61511). He has 25+ years of experience in safety and critical control systems in process control SIS, railways systems and medical devices, including product development, project engineering, project management and technical product management.

Contact us

For feedback and comments about the content of this white paper:

Triconex Safety Validator
Real-time-answers.com

If you are a customer and have questions specific to your application logic testing requirements:

Contact your Schneider Electric representative at
http://www2.schneider-electric.com/sites/corporate/en/support/operations/local-operations/local-operations.page

Appendix A: References

IEC6108: Edition 2.0 Functional safety of electrical/electronic/programmable electronic safety-related systems

IEC61511: Edition 2.0 Functional safety - Safety instrumented systems for the process industry sector

HSE Human Factors Briefing Note 10 – Fatigue
http://www.hse.gov.uk/humanfactors/topics/10fatigue.pdf

Testing of application logic in the safety logic solver – Ajay Mishra
12th International TÜV Rheinland Symposium