

EMC Test Laboratory Software Program Development: A Proven Process

An EMC test engineer's primary responsibility includes creating test plans, procedures, analyzing test data, generating test reports, and mitigating noncompliance challenges. Many times the EMC engineer must assume the role of a software test engineer and create test programs necessary to meet the project objectives. The EMC engineer may or may not have the intrinsic knowledge to create test programs while meeting laboratory requirements (i.e. ISO-17025[1]). In addition, Laboratory accreditation auditors require the laboratories to provide evidence their test software has been verified and validated[2]. This article was written to give EMC test personnel guidance in creating their test programs, while verifying and validating their test software within the development cycle in order to expedite test throughput and meet laboratory requirements.

BY JACK MCFADDEN

I have been in the testing field for over twenty five years. I started in the Department of Defense (DOD) world. I gained experience in the automotive industry and fine-tuned my skills through a mixture of the DOD, automotive and commercial industries. I learned to create test programs through necessity, not desire. For the majority of my career I worked within an organization's internal laboratory. Our true software test engineers

were focused on production. I had tremendous difficulty enlisting their support. Our senior management had this strange idea. They believed the software engineers' time was best spent where we received a positive return. So the internal testing laboratories were left to fend for themselves. The result was that I learned to create test programs on my own. Naturally, I made just about every mistake you could imagine along the way, but I grew

and learned from them. I know and understand there can be a difference in what you intend for the computer to do versus what the computer actually does and that the measure of a competent test software engineer is his or her ability in creating test programs that perform what he or she intends. I have struggled over the years with getting the system to do what I needed it to do rather than what I inadvertently told it to do. This article is designed to



improve your probability of developing successful test programs. It can help you teach your test system and instruments to behave themselves.

TEST PROGRAM DEVELOPMENT METHODS

When I began designing test programs I used the classic design approach which is commonly called the Waterfall^[3] Method as illustrated in Figure 1. The Waterfall Method starts with the requirements, moves straight into design, then implementation, followed by verification and ends in maintenance. Its biggest drawback is the lack of feedback. You have completed the majority of the development process before you can verify the test program. This can lead to costly errors as you try to navigate back upstream. I knew there was a better way. So I spent some time researching. I found other development methods. I learned how I could reduce risk and increase my probability of first time success. The software development process I prefer is called the Revised “V” Cycle^[3]. It was developed in the 1980s. The Revised “V” Cycle was an evolution of the “V” Cycle and the “V” Cycle was the evolution of the Waterfall Method. The Revised “V” Cycle embeds feedback into every phase of the development process. This feedback enables the design to be modified in the program early within the development cycle. I found it is far easier and less costly to correct issues in the very beginning of the development process. Early fault detection also reduces your pain as the project heads downstream. So the old adage *test early, test often* is critical to developing a useable program. The Revised “V” Cycle process is shown in the following figure. If you want more information regarding the different software development methods please visit: http://www.aiglu.org/aiglu_documentations/agile-introduction.

As shown in Figure 2, the Revised “V” Cycle component phases are

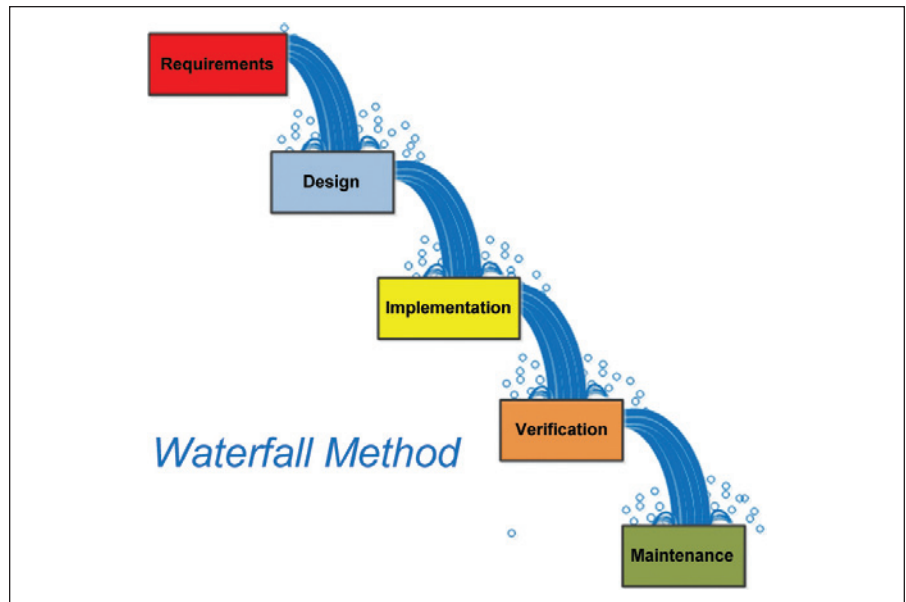


Figure 1: Illustration of step down waterfall method.

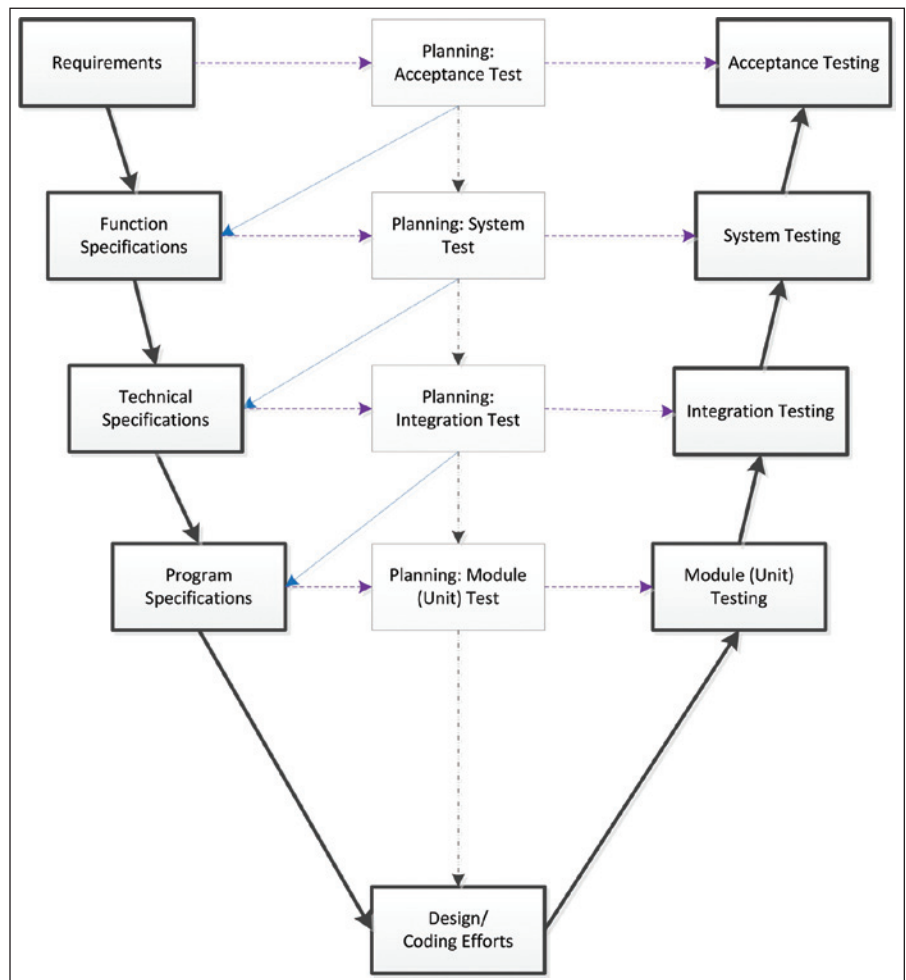


Figure 2: Revised “V” Cycle software development process showing interdependence of component phases.

interdependent on each other. You start with the requirements, specification (function and technical) as well as the program specifications while beginning the Planning: Acceptance, System, Integration and Module phases. You move from one phase into another while feeding back the information you discovered during the previous phase.

PROGRAM DEVELOPMENT PROCESS (PDP)

As you can see, the test program development is a process. There are specific tasks that are repeated over and over. Understanding the

process enables you to break it down into its discrete components and standardize them. Standardizing the process benefits everyone involved. The standardized process objective is development method tools that reduce risk. For example, if you develop a process that is dependent on highly skilled personnel, your process could be impacted if the highly skilled personnel had a bad day. Unintentional errors could be injected into the process which could invalidate the results and end up costing you time and money. Creating standardized processes can help decrease the potential errors or at the very least identify them and

implement corrective action. The test program development process^[4] I use is shown in Figure 3.

PDP - Test Requirements

The first step is to understand your requirements. The test standard has specific rules. The standard rules are clear. They are the “shall” statements found within the standards. The test specimen must comply with specified limits, whether below an emission predetermined level or for immunity, above a certain level. As an example, I am using the MIL-STD-461F^[5], Radiated Emissions (RE)102.

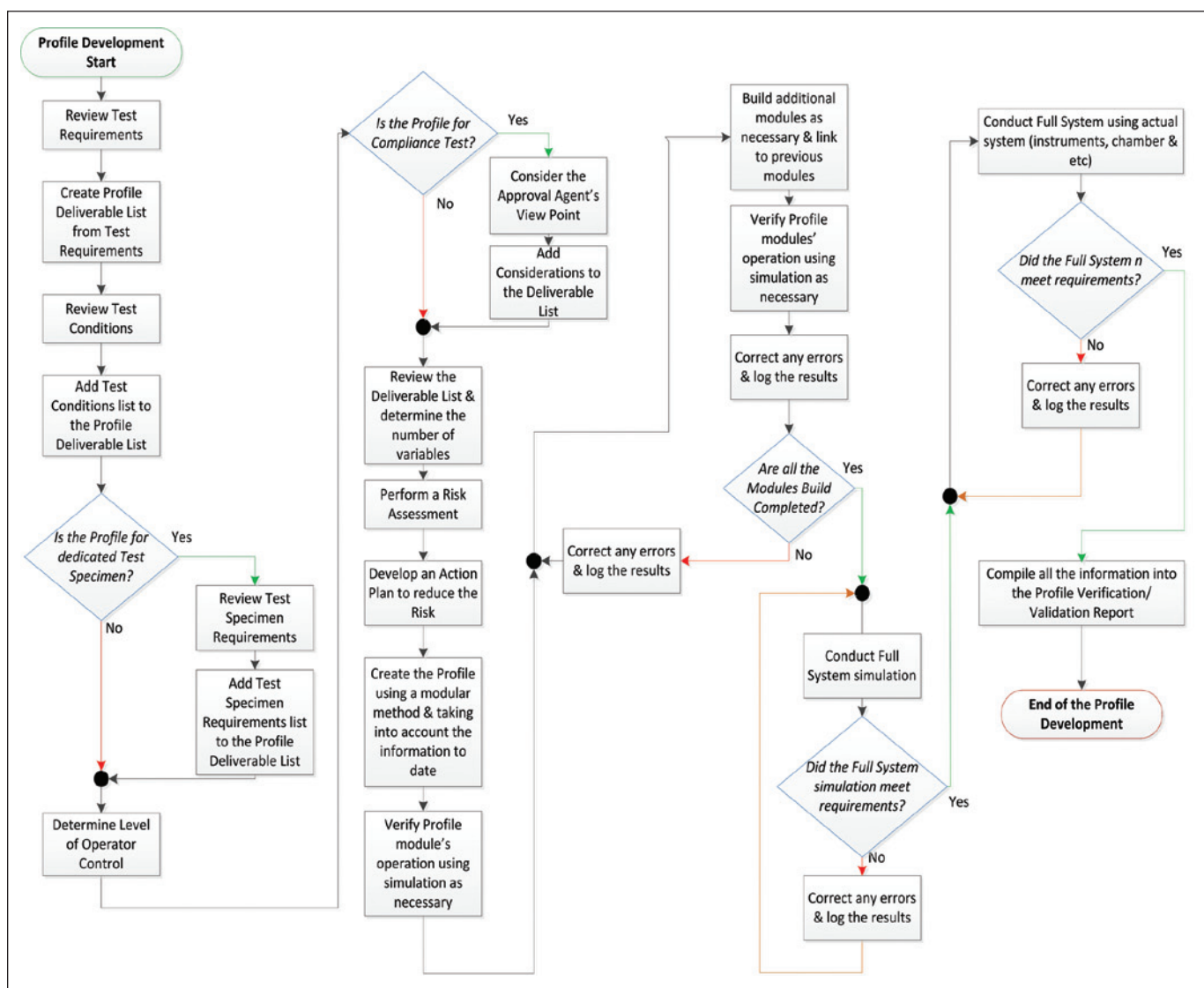


Figure 3: Flowchart showing software program development process used by the author.

The test requirement information is described in Table 1 and Figure 4. I will use the term equipment under test (EUT) for the test specimen when discussing the MIL-STD example since the MIL-STD uses EUT as the definition of test specimen.

Initially, you might argue the requirement is actually the “EUT Active Limit” as shown in Figure 4. This is partially true. The final objective is to measure the EUT emissions. The EUT emissions must be lower than the applicable limit for the EUT to comply

with the MIL-STD; however, the test laboratory is required to prove their test system and ambient conditions are capable of making the final (EUT Active) measurements. It is the requirement of the laboratory. It needs to be addressed in the beginning of the

Item Description		Requirements	Tolerance
Equipment Under Test (EUT) Active Limit		The peak corrected levels shall not exceed levels shown in Test Requirements Graph.	±3 dB
Ambient Limit		-6 dB below EUT Active Limits, Test Requirements Graph. Results are to be included within the test report if Active Mode results exceed Limits.	
System Check	Continuity	Verify signal path using a calibrated signal at low, medium and high frequencies of a rod monopole and a calibrated signal at the highest frequencies of the remaining antennas. The rod system verification use a 10 pf capacitive network within the signal path as described within MIL-STD-461F. This 10 pf capacitive network cannot be a commercial product.	
	Stub Radiator	Verify antenna (signal) path's integrity at each of the antenna's highest frequency.	

Table 1: MIL-STD-461F, RE102 Test Requirements

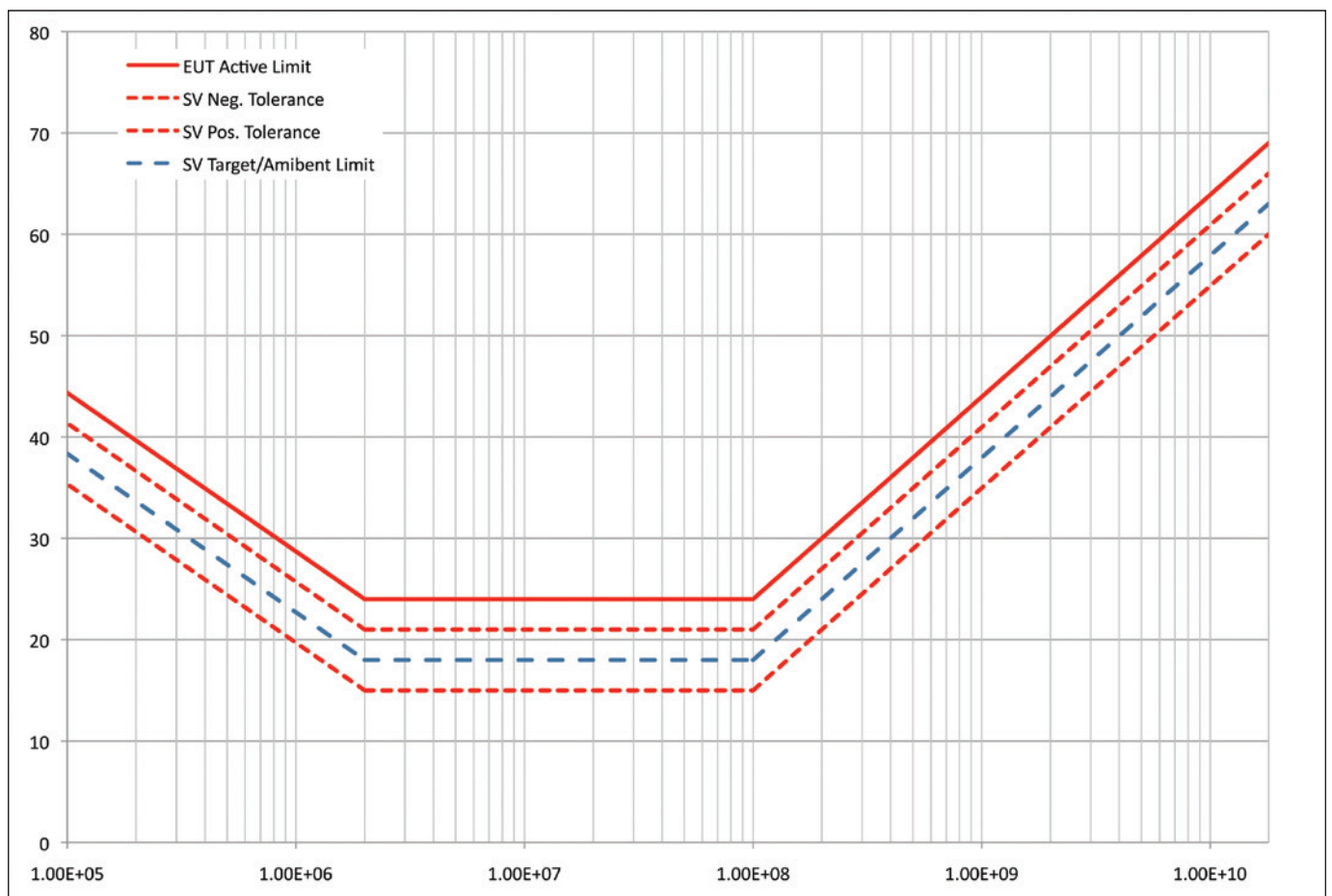


Figure 4: MIL-STD-461F, RE102 Test Requirements Graph

test program development. If you are satisfied that you have considered the test requirements, I highly recommend documenting them within a deliverable list in order to ensure they are not missed.

PDP - Test Conditions

This moves us to the next step of the process. What are the test conditions? Test conditions are the “should” statements within the standard. These are the test parameters necessary to meet the standard, but there is an element of choice within the standard parameters.

Test conditions answer the following questions: How fast should the frequency sweep be made? What is the frequency resolution, etc.? Table 2

continues the MIL-STD-461F, RE102 example.

Note the antenna beamwidth is more of a test setup condition. It typically does not enter the test program arena unless you are one of the fortunate laboratories that possess a remote positioner. If you have a remote positioner, this should be included within your test program. If you do not have a remote positioner, then you need to decide if you are going to provide guidance to test personnel on the correct test setup or not. The information you gathered should be entered into your deliverable list.

PDP – EUT Monitoring Requirements

This brings us to the next step in the process. It is the first decision block

within the flowchart. It is regarding where the EUT operation monitoring duties fall. There are times when the actual EUT is monitored independently from the test program. There are other times when the EUT’s operation is integrated within the test program. As shown in Table 3 (page 6), you need to determine if the test program has EUT monitoring requirements. At this point I will go back to the MIL-STD-461F, RE102 example. You should answer the following questions: Does the test program need to accommodate the monitoring of the EUT operation? If you are required to monitor the EUT, then how are you to measure and record the EUT’s performance? How susceptible is the monitoring and support equipment? What is the monitoring and support equipment’s emanation levels? If you don’t know the

Test Condition I.D.	Test Condition Requirement			Units
Test Standard	MIL-STD-461F			n/a
Test Type	Emissions			n/a
Measurement Instrument	Receiver			n/a
Detector/Measurement Method	Peak			n/a
Frequency Accuracy	2			%
Sweep Method	Single Sweep			each
Measurement Tolerance	±3			dB
Resolution Bandwidth	Frequency Range		Min. Duration seconds	Min. Number of Data Points
	Start	Stop		
1 kHz	10 kHz	150 kHz	4.2	280
10 kHz	150 kHz	30 MHz	89.55	5970
100 kHz	30 MHz	1 GHz	291	19400
1 MHz	1 GHz	18 GHz	510	34000
Video Bandwidth	Maximum available			
Antenna Beamwidth Requirements				
Active Monopole Rod	Multiple position required if test boundary exceeds 3 meters			
Biconical				
Large Double Ridged Horn 69 x 94.5 cm opening	EUT width plus 35 cm of cable harness			
Small Double Ridged Horn 24.2 x 13.6 cm opening	EUT width plus 7 cm of cable harness			

Table 2: MIL-STD-461F, RE102 Test Conditions

emission and immunity characteristics of the EUT monitoring/support equipment, you may find yourself generating more work than necessary during actual testing. You also need to understand what the acceptable tolerance of the EUT is. In other words, when does the EUT operation go from an acceptable condition to an unacceptable condition? For example, if you are testing a video product, when does its normal operation cease and it becomes susceptible? Is it when there are little transparent disturbance horizontal/vertical lines observed on the video, but it otherwise remains legible, or is it when you can no

longer read the print or understand the graphics? You must be extremely careful if the EUT susceptibility is defined with “any deviation” as then any EUT normal variation could be interpreted as a susceptibility response. The MIL-STD covers this by stating words to the effect that EUT susceptibility response is any deviation greater than the specified parameters found within the product specification. Once you know the answers, you should record them into the deliverable list which ends this part of the process. This discussion is summarized in Table 3, EUT Monitoring Characteristics.

PDP – Test Personnel

The next item to consider is your test personnel. What type of control will you give your test personnel during testing? If you have test personnel with an expert skill level, you might consider relaxing the number of prompts and controls for the test program. If your test personnel have a novice skill level, then you might want stricter controls and more operator prompts guiding the test personnel throughout the course of the test sequence(s). The evaluation of test personnel and control level is shown in Table 4. It is far safer to develop a test program with the novice in mind than the expert. After you

EUT Monitoring	Applicable	Not Applicable
EUT Acceptance Criteria	Product specification allowable tolerance	
EUT Susceptibility Criteria	How to determine if the EUT is susceptible?	
EUT Support Equipment Susceptibility	How susceptible or immune is the EUT support equipment?	
	What is the EUT monitoring/support equipment contribution to the ambient conditions?	
EUT Monitoring Method	Integrated within test program	Independent of test program

Table 3: EUT Monitoring Characteristics

Test Personnel	Novice	Nominal	Expert
Test Control Level	High	Medium	Low

Table 4: Evaluation of Test Personnel and Control Level

Test Level	Pre-compliance (Evaluation, R&D)	Compliance (Qualification)

Table 5: Determining the Test Level for the Program

Item I.D. No.	Risk Description	Item I.D. No.	Risk Description
1	Incorrect frequency range	2	Incorrect resolution bandwidth
3	Incorrect sweep time	4	Incorrect number of data points/steps
5	Incorrect video bandwidth	6	Incorrect detector
7	Missing/incorrect data plots	8	Incorrect equations
9	Incorrect/missing transducer factors	10	Incorrect/missing data arrays
11	Incorrect switch settings	12	Incorrect limit
13	Incorrect instrument drivers	14	Incorrect instrument addressed
15	Incorrect pre-selector settings	16	Incorrect preamplifier settings

Table 6: Possible Causes of Errors in MIL-STD-461F, RE102 Test Program

know your answer, you should record it into the deliverable list.

PDP – Test Type

This moves us into the next phase of planning action. What is the test type? Is it compliance testing, pre-compliance, or research and development (R&D)? The requirements for compliance testing are the most severe while pre-compliance and R&D can be less stringent depending on the objective. For example, you could reduce the system checks for a pre-compliance evaluation or reduce the frequency range to a specific area of interest. Either way, as shown in Table 5, the test program needs to account for the test type and again it is far easier to go with the most stringent level than reduce the scope and increase the test scope at a later date and time.

PDP- Risk Analysis

You have almost completed the fact gathering of the process. There is one other item to consider then it is time to analyze the information you compiled. This is one of the most important phases of this process. You know what you need. It is covered in your deliverables list. Now you have to understand what can stop you from achieving your objective. I have experienced many times a test program failure due to an unaccounted for or incorrectly set variable. So this is the time to determine how many different variables you need to accommodate and control. The more variables there are, the greater the chance for error. After identifying the potential error causes it is time to conduct a risk assessment. The results of a risk assessment^[1, 2 & 3] should include a corrective action for any score

determined to be a medium to high risk.

Going back to the MIL-STD-461F, RE102 example, I have determined sixteen possible test program error causes and listed them in Table 6.

The next step is to create the risk assessment scoring. There is a risk scoring method described within the Software Quality Engineer's (SQE) training manual^[6]. If you do not have access to the SQE training manual then I recommend using a standard risk assessment process: identify the risk, ascertain the risk score, summarize the risk characteristics, and create corrective actions in order to reduce the risk score, assign resources and create a risk analysis database for tracking purposes. Table 7 reflects the end result of the risk assessment from the MIL-STD-461F, RE102 example.

Item I.D. No.	Description	Likelihood of Occurrence	Severity/ Impact	Score	Correction Action
1	Incorrect frequency range	Medium	High	High	SR & PS
2	Incorrect resolution bandwidth	Medium	High	High	SR & PS
3	Incorrect sweep time	Medium	High	High	SR & PS
4	Incorrect number of data points/steps	Medium	Low	Medium	SR & PS
5	Incorrect video bandwidth	Medium	Low	Medium	SR & PS
6	Incorrect detector	Low	High	Medium	SR & PS
7	Missing/incorrect data plots	Low	High	Medium	SR & PS
8	Incorrect equations	Medium	High	High	SR & PS
9	Incorrect/missing transducer factors	Medium	High	High	SR, PS & PE
10	Incorrect/missing data arrays	Medium	High	High	SR & PS
11	Incorrect switch settings	Low	High	Medium	SR & PE
12	Incorrect limit	Low	High	Medium	SR & PS
13	Incorrect instrument drivers	Low	High	Medium	SR & PE
14	Incorrect instrument addressed	Low	High	Medium	SR & PE
15	Incorrect pre-selector settings	Low	High	Medium	SR & PE
16	Incorrect preamplifier settings	Low	High	Medium	SR & PE
Legend:					
SR	Static Review				
PS	Performing program operation using Simulator (if available – if not available, use PE)				
PE	Performing program operation using actual Equipment (system and acceptance testing)				

Table 7: End Result of the Risk Assessment from the MIL-STD-461, RE102 Test Program

Test Type	Test Condition	Description		Results
V/V Modular	Modular testing discretely verifies individual program components			
	Zero Fault	Limit Select	Limit Calculations	
	Intentional Fault(s)			
	Zero Fault		Frequency Range Calculations	
	Intentional Fault(s)			
	Zero Fault	Instruments	Select Instrument Drivers	
	Intentional Fault(s)			
	Zero Fault		Instrument Addresses	
	Intentional Fault(s)		Instrument Addresses	
	Zero Fault	Transducers	Loading Transducer Correction Factors (antennas, signal path, preamplifiers and etc.)	
	Intentional Fault(s)			
	Zero Fault	System Check Sweep	Resolution Bandwidth	
			Video Bandwidth	
	Zero Fault	System Check Sweep	Frequency Resolution	
			Sweep Time	
			Detector Setting	
			Number of Data Points	
			Calculations	
			Switch Settings	
	Zero Fault	Ambient Sweep	Pre-selector Settings	
			Receiver Settings	
			Preamplifier Settings	
			Resolution Bandwidth	
			Video Bandwidth	
			Frequency Resolution	
			Sweep Time	
			Detector Setting	
			Number of Data Points	
			Calculations	
			Switch Settings	
	Zero Fault	Test Active Sweep	Pre-selector Settings	
			Receiver Settings	
			Preamplifier Settings	
			Resolution Bandwidth	
			Video Bandwidth	
			Frequency Resolution	
			Sweep Time	

Test Type	Test Condition	Description		Results
V/V Modular	Zero Fault	Test Active Sweep	Detector Setting	
			Number of Data Points	
			Calculations	
			Switch Settings	
			Pre-selector Settings	
			Receiver Settings	
			Preamplifier Settings	
	Zero Fault	Report	Labeling	
	Intentional Fault(s)			
	Zero Fault		Generation	
Intentional Fault(s)				
Zero Fault	Data Plots and Tables			
V/V System Integration using simulation	Integration checks the full operation (transducers, instruments, calculations, report and etc.) of specific actions			
	Zero Fault	System Check, calibrated signal source(s)		
	Intentional Fault(s)			
	Zero Fault	Ambient Sweep, known test conditions		
	Intentional Fault(s)			
	Zero Fault	Test Active Sweep, known test conditions		
	Intentional Fault(s)			
V/V System Testing using actual instruments	System tests the full operation (transducers, instruments, calculations, report and etc.) of specific actions			
	Zero Fault	System Check, calibrated signal source(s)		
	Intentional Fault(s)			
	Zero Fault	Ambient Sweep, known test conditions		
	Intentional Fault(s)			
	Zero Fault	Test Active Sweep, known test conditions		
	Intentional Fault(s)			
V/V Acceptance Testing	Acceptance tests full operation (transducers, instruments, calculations, report and etc.) of specific actions and usually enlists third party operation/witness			
	Zero Fault	System Check, calibrated signal source(s)		
	Intentional Fault(s)			
	Zero Fault	Ambient Sweep, known test conditions		
	Intentional Fault(s)			
	Zero Fault	Test Active Sweep, known test conditions		
	Intentional Fault(s)			

Table 8: The Verification/Validation Phase for a MIL-STD-461F, RE102 Test

Adherence to a test laboratory development software process described will generate evidence needed for internal and external quality audits and provide greater confidence that your test programs are doing what you intended them to do. The result is expedited test throughput while meeting laboratory requirements.

PDP – Functional Modules Building and Links

The results of the process so far have generated a deliverables list and risk assessment table/database. It is time to start building the program using the information you have created. I highly recommend building the test program using a modular method and running the modules operation at specific stages in order to keep to the adage: *test early, test often*. Using the MIL-STD-461F, RE102 example, you can break the test program into discrete components: limit select, instruments configuration, transducer correction factors, system check, ambient sweep, test sweep, report labeling, and report generation. If you record the results of the modular build testing, you are building documentation you can use for your

test program verification/validation report. You continue this process until all modules have been created, debugged and linked. After you have completed this phase you move from code creation to testing (verification/validation). Please note, debugging is part of the development process. It does not start the verification/validation process^[3].

I recommend generating a test case table where you enact the entire operation of the test program^[2 & 3]. You build into the test case both zero user fault conditions and intentional user fault conditions then observe how the test program responds to the conditions. User fault conditions are actions where the user could enter or generate unintentional errors. Here

are a few fault examples: The user could incorrectly select the limit. The user could load the wrong transducer corrections factors. The user could load an incorrect instrument driver. Where the operator or user does not have input, I recommend using zero fault conditions only. I see little value injecting errors into the system where or when there is little probability of that error occurring during actual operation. The intent is to find and eliminate potential errors. It is not to embed them into the system.

PDP – Verification/Validation (V/V)


Start the verification/validation phase at the lowest level (modular) then increase the scope until you have a full system acceptance test. If the test

The standards/documents mentioned in this article include:

I.D. Number	Title	Rev.	Date
1	ISO/IEC 17025 General Requirements for the Competence of Testing and Calibration Laboratories, International Organization for Standardization (ISO)	2	2005
2	Software Validation in Accredited Laboratories, A Practical Guide, Gregory D. Gogates, FASOR Inc. ftp://ftp.fasor.com/pub/iso25/validation/adequate_for_use.pdf	n/a	07 June 2010
3	Software Training and Consulting (SQE Training) Testing, Development, Management Requirements and Security	V4.1	2004-2011
4	ETS-Lindgren's TILE Profile Development Process, J. McFadden http://support.ets-lindgren.com/TILE	n/a	2012
5	MIL-STD-461F, Requirements for the Control of Electromagnetic Interference Characteristics of Subsystems and Equipment, Department of Defense Interface Standard	F	10 December 2007
6	Description of the SWEBOK Knowledge Area Software Engineering Process (Version 0.9), Khaled El Emam, National Research Council of Canada, Institute for Information Technology NRC, Canada	0.9	2001

results indicate a fault, then corrective action should be performed immediately prior to moving to the next phase. As always, record the results to build evidence for your verification/validation report. An MIL-STD-461F, RE102 example is shown in Table 8.

CONCLUSION

The article followed a software development process using the Revised “V” Cycle. Adherence to a test laboratory development software process^[2] described will generate evidence needed for internal and external quality audits and provide greater confidence that your test programs are doing what you intended them to do. The result is expedited test throughput while meeting laboratory requirements. 

ACKNOWLEDGEMENT

The author would like to recognize and thank David R. Guzman of RFTek for his helpful review of this article.

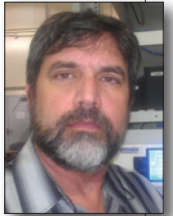
REFERENCES

More information on software quality engineering can be found at

- American Society for Quality
www.asq.org
- American Software Test Qualification Board, Inc.
www.astqb.org
- Society Quality Engineering
www.sqe.org
- ETS-Lindgren TILE Support
<https://support.ets-lindgren.com/TILE/>

(the author)

JACK MCFADDEN is an EMC Systems Engineer with ETS-Lindgren in Cedar Park, Texas. His responsibilities include EMC test system design and integration. Prior to joining ETS-Lindgren, he worked for Wyle Laboratories where he was a Senior Project Engineer. Mr. McFadden is an iNARTE certified EMC engineer as well as an iNARTE certified EMC technician with over 25 years experience in EMC test systems and software development. He is a certified tester foundation level (CTFL) per the American Software Testing Qualifications Board, Inc. (ASTQB). He may be reached at Jack.McFadden@ets-lindgren.com.



If you're reading this article, chances are you need compliance related information to do your job and do it well. Fortunately, there's more where this came from. As a subscriber you have access to our monthly print publication as well as a wealth of information on line. Subscriptions are free to qualified industry professionals.

Subscribe online at incompliancemag.com/subscribe

IN COMPLIANCE
Magazine