Solving Tomorrow's Obsolescence Management Challenges with System Design

CONTENTS

Focus on Sustainment and Obsolescence in Operational Decision Making 2
- Choose Reliable Platforms and Vendors 2
- Cooperate with Your Vendor 3
- Cooperate with Procurement 4
- Document an Obsolescence Management Plan 4

Hardware Selection and Integration 5
- COTS Components 5
- Open Industry Standard 5
- Plug-In, Modular Hardware Architecture 6
- Synthetic or Software-Defined Instrumentation 6
- Hardware Platform Investment and Support 7

Software Tools and Architecture 8
- COTS Software Tools 8
- Test Software Toolchain and Interoperability 8
- Modular Software Architecture 9
- Functional Abstraction Layers 9
- Technical Support and Training 10

Next Steps 10

Test systems built to manufacture and support aerospace and defense platforms generally need to remain in service for the lifetime of that platform, or at least long enough to perform planned sustainment over 20 or 30 years. Most test systems are not built in a way that includes sustainment engineering as part of the initial design.

Consider using these best practices in operations implementation, hardware acquisition, and software design to reduce the sustainment burden of handling obsolescence in test systems long before the equipment goes end of life. Following these best practices can cut the number of engineering hours by half, or more, and reduce associated costs by hundreds of thousands, or even millions, of dollars over the lifetime of the test system.

Focus on Sustainment and Obsolescence in Operational Decision Making

Designing a test system for long-life operation means making decisions with the entire life cycle of the system in mind.

Choose Reliable Platforms and Vendors

One of the ways you can improve your obsolescence-readiness decision-making process is to choose the right lifetime-optimized products and vendors who emphasize engineering for a long-life cycle.

Two high-level best practices in choosing building blocks for your long-life test systems are to work with commercial off-the-shelf (COTS) tools and use industry-standard platforms that are managed by multiple vendors and end users. Several government associations, test and measurement industry committees, and private organizations are working toward standardized and interoperable platforms with many suppliers. Some examples are the Sensor Open Systems Architecture (SOSA) and PXI Systems Alliance (PXISA). Purchasing from vendors who cooperate with these organizations ensures that the platforms you use have been vetted and offer multiple options for long-term sustainment.

Some engineering tools vendors have policies, services, and cooperative engagements. These policies can extend all the way to new product development. For instance, new PXI products and instruments developed by NI must support multiple releases of LabVIEW and maintain operational continuity from version to version. The PXI-4060 model of the PXI Digital Multimeter (DMM) that was introduced in 1998 uses the same "Fetch" function for measurements in the NI-DMM instrument driver as the PXIe-4081 DMM that was released in 2017.

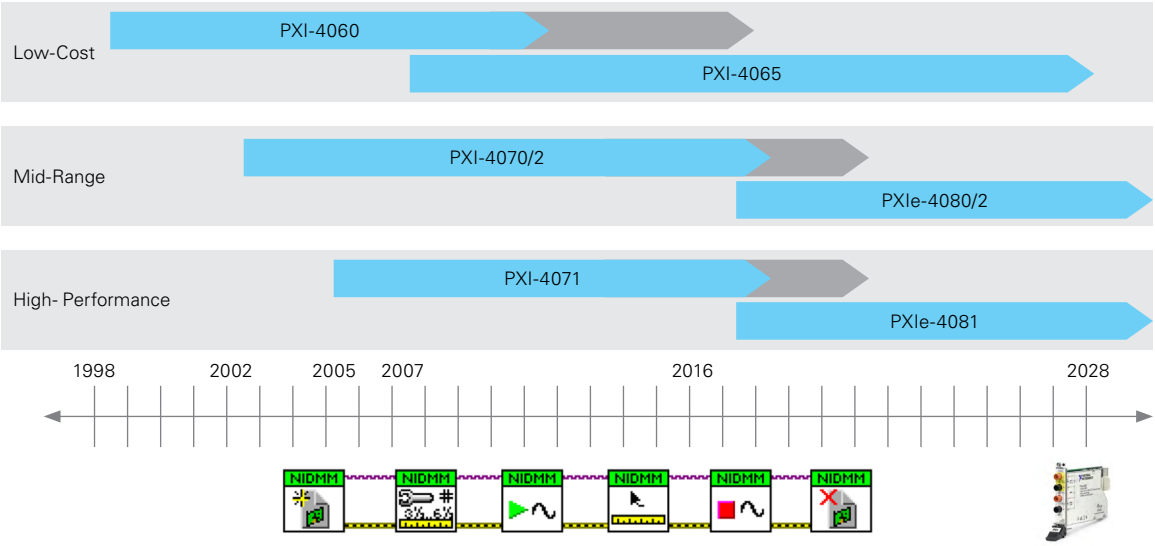


Figure 1. NI DMMs have used the same driver functions since the first one was released. The PXIe-4081 DMM operates with the same code that was written to work with the PXI-4060 in 1998.

Cooperate with Your Vendor

The best long-term test systems are built on platforms that feature sustainment plans continually updated with all the essential life-cycle information for the system's hardware components. Obtaining life-cycle information requires establishing a cooperative relationship and good communication with suppliers. It also requires diligent suppliers who create plans. Instrument vendors should empower you to plan for technology evolution in your system, even sharing roadmap information where possible. They should also provide services ranging from up-front consulting on product selection to long-term extended service agreements to meet your specific needs.

If you have decades-old products like the PXI-4060 DMM in your test system, NI is happy to regularly engage in a life-cycle review of that test system to measure the risk of obsolescence of each instrument and consider timelines for technology insertions. See Figure 2 for an example of a technology life-cycle review. By engaging in reviews like this, you can plan a single technology insertion project to replace multiple aging components at the same time. This reduces engineering effort and cost and prevents unforeseen obsolescence events. Then, instead of fighting fires, you can properly plan the headcount and budget you need to refresh technology and investigate new products and capabilities from the vendor to extend the features of your test systems.

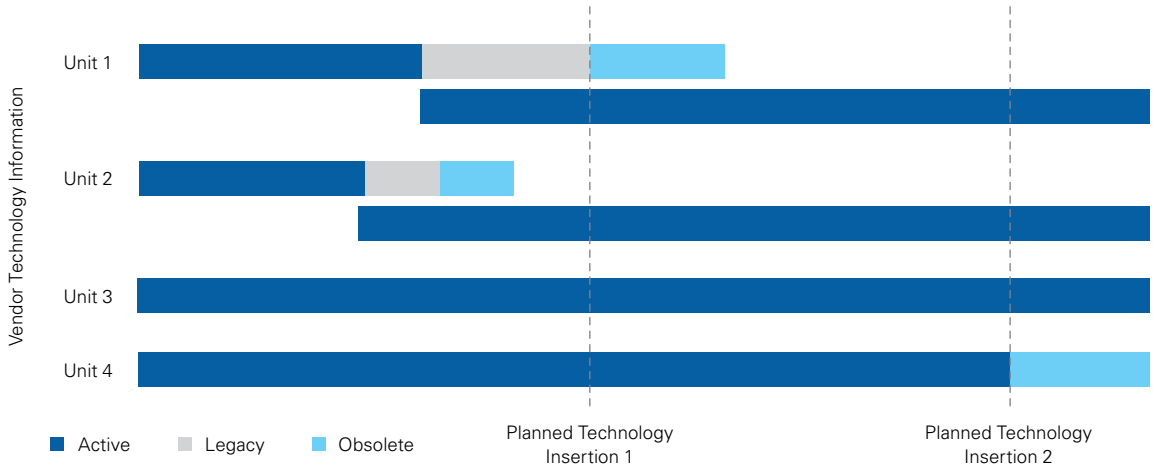


Figure 2. This is an example life-cycle review and the resulting plan for replacing instrumentation with more modern options. Planning technology insertions with your tools vendors reduces the downtime risk of your test system.

The following sections on hardware and software best practices explore how to choose new products to build into your test systems. They also discuss how to select the vendors for those new products.

> "CACI's relationship with NI has grown to a level of mutual trust as we work together to deliver high-quality, sustainable test solutions at affordable prices."
>
> Paul Pankratz, CACI

Cooperate with Procurement

Once you have selected the right products and vendors, you can improve your decision-making operation by involving your acquisition team in defining the system. If you can identify critical equipment or vendor features that will save money on other tools and engineering effort over the lifetime of the system, you can work with the purchasing team to include those as requirements in the final system. That significantly simplifies the proposal and purchasing process.

Document an Obsolescence Management Plan

The last operational step to improve sustainability is to implement an obsolescence management plan in the documentation of the system at the time of delivery. This plan should offer information for replacing all system components including when they should be replaced, how critical each component is to the operation of the test system, and how much risk is introduced by the obsolescence or replacement of that component. The criticality and risk are the most important pieces of information to capture in this plan. The team designing the test system often knows far more about these issues than those maintaining the system 20 years later and can explain the effort needed to replace each component. See Figure 3 for an example obsolescence management plan.

Component	Plan of Record	Replacement Component	Timing	Criticality and Risk
Custom Cable	Drop-In Replacement (Vendor-Supplied)	Custom Cable	Immediate	Medium and Low
1 kΩ Resistor Network	Drop-In Replacement (Vendor-Supplied)	1 kΩ Resistor Network	Immediate	Low and Low
Racal 4152A DMM	Replace with Similar (Vendor-Supplied)	NI PXIe-4080 DMM	Technology Refresh	Medium and Medium
Windows XP	Replace with New (Vendor-Supplied)	Windows 10	Technology Refresh	High and Medium
Virtex-2 FPGA	Last Time Buy	N/A	N/A	High and Low

Figure 3. A test system should have an obsolescence plan that describes how to handle the end of life for any component in the system. The plan should list all the factors that contributed to that decision including the criticality and risk of that component becoming obsolete.

Hardware Selection and Integration

You should always consider the sustainability of each piece of hardware you select when designing a new test system to operate for decades. Just as important, however, are the vendor services for sustaining the system and the skills needed to operate and maintain it over time.

COTS Components

The most sustainable test systems are built with COTS components. Using COTS products expands the user base for a given product, which improves the likelihood that the product will be properly maintained. Test hardware from large vendors has many users, so updates to firmware, drivers, and the hardware life cycle are planned carefully to reduce impact across a wide user base.

Open Industry Standard

To ensure that your test system can be sustained over decades, you need to select a hardware platform that is continually growing as well. Then you can avoid completely redesigning a test system architecture when a single component goes end of life or when you need new measurement capabilities. Open industry platforms such as PXI, VXI, and GPIB deliver the benefits of multiple vendors who are innovating on the hardware platforms through instrumentation hardware competition. This competition fosters healthy platform growth and a constant supply of new products to meet the needs of advanced test systems.

The PXISA includes more than 70 test and measurement vendors who oversee the maintenance and innovation of the PXI test hardware platform, which is an open-standard platform. Growth of the PXI platform has been rapid since the adoption of the standard in 1998. In addition to a vast product offering, PXI is expected to continually grow for the foreseeable future, as shown in Figure 4, making it an ideal platform for use in long-term test systems.

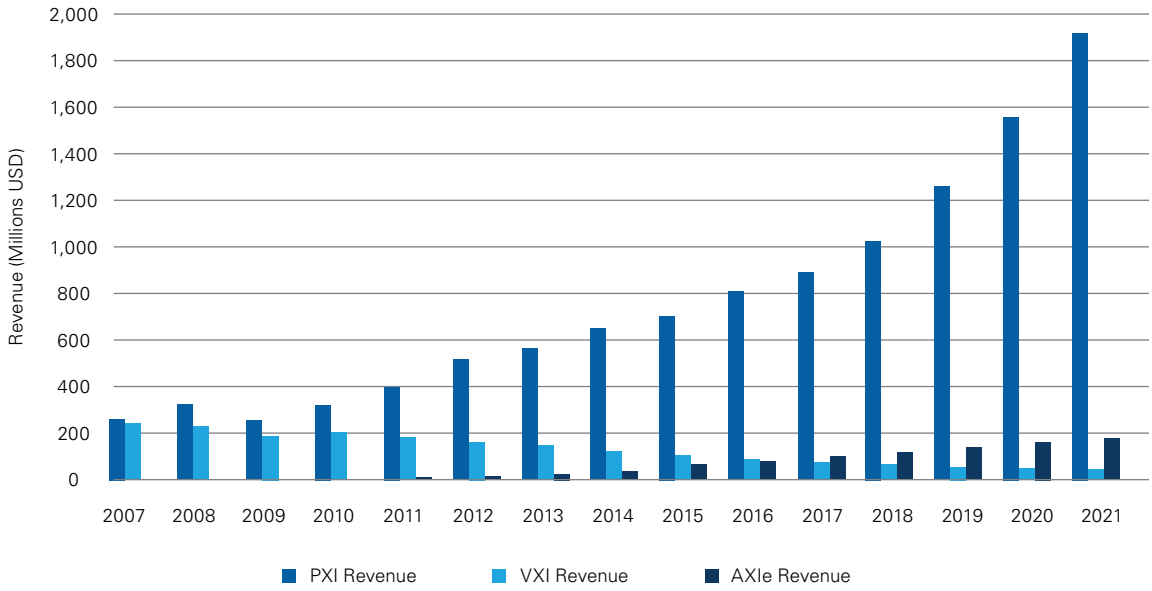


Figure 4. PXI is the dominant plug-in test instrumentation platform on the market today. Experts project continued growth for PXI over other modular test instrumentation platforms for the foreseeable future.

Plug-In, Modular Hardware Architecture

Open standards that have modular, plug-in components further decrease system costs by maximizing component reuse and reducing technology insertion effort. Replacing a traditional instrument means accounting for size, heat generation, power consumption, and other factors. Upgrading or replacing a modular instrument is as easy as removing the old instrument from its slot in the carrier and replacing it with the new one.

Plug-in architectures also simplify test system expansion. Test systems built for longevity are often required to incorporate more I/O over time to test new features or line replaceable units (LRUs). Having test systems that can stand the test of time requires an instrumentation platform with a large portfolio of products that can perform tests on DC, analog, digital, and RF signals at various levels with accuracy and speed.

Synthetic or Software-Defined Instrumentation

Test hardware in modern test systems often needs to perform many measurements and tests. Instruments with software-configured measurements have the flexibility to take the right measurement and get the results needed. Many times, these instruments have extensive code compatibility with other instruments using industry-standard APIs, like VISA or IVI, or well-engineered vendor-defined APIs. Combining the right complementary hardware and software tools can facilitate the design of long-term test system architectures.

Instruments with open FPGAs add another level of compatibility by helping you design the firmware of an instrument and reuse that firmware on compatible instruments as necessary. This type of customization isn't always necessary, but it can help promote important features that could go obsolete over time.

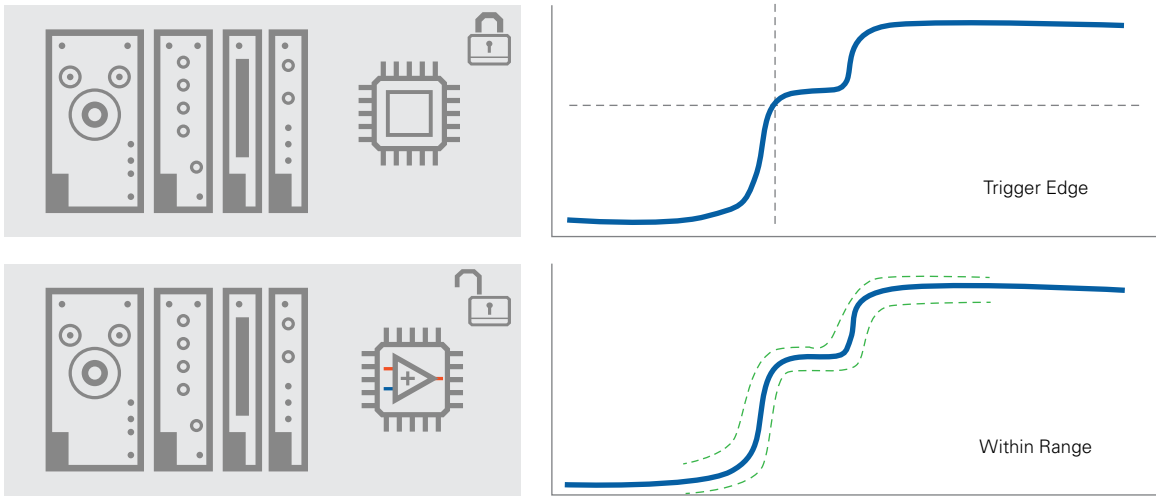
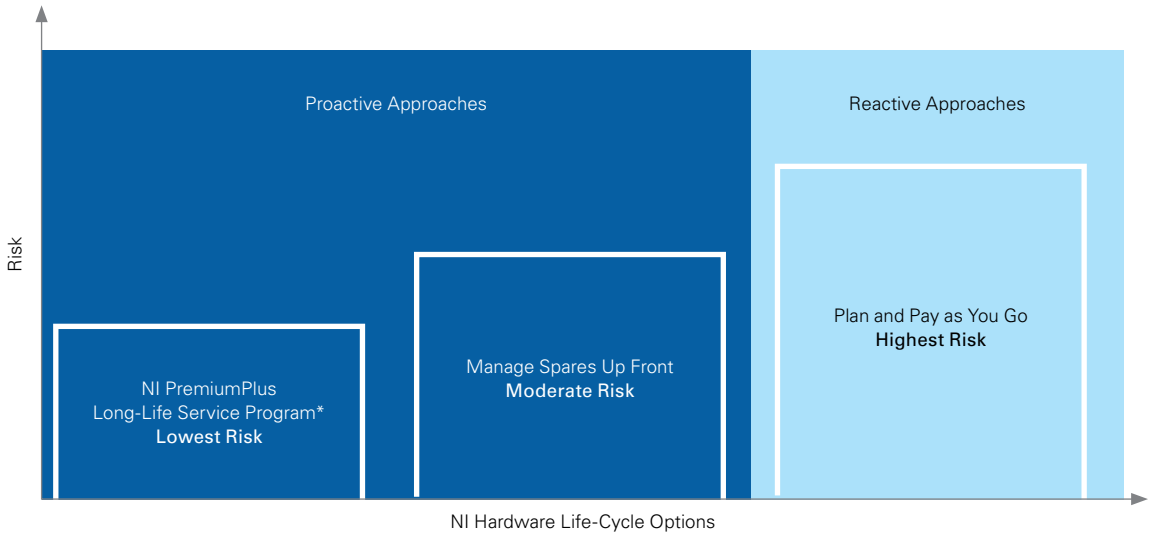


Figure 5. NI devices with programmable FPGAs help you create custom measurement features like triggers and signal processing as well as interoperative device firmware.

Hardware Platform Investment and Support

Another sustainability step is to ensure that test products are manufactured by vendors who have strong track records for continual bug fixes and software support updates as well as the ability to provide replacement products for aging ones. Without vendor support, you may have difficulty troubleshooting technical issues and unforeseeable problems in the system.

A good hardware vendor also provides options for warranties on instrument function, calibration services and instructions, repair plans, and even the ability to reserve spare hardware to ensure minimal downtime if instruments in the test system break down. NI offers several levels of service contracts, from coverage on a single instrument to system-wide coverage that lasts up to 20 years.



*Available durations depend on hardware life-cycle phase at the time of purchase.

Figure 6. Collaborate with NI on a service program that extends the life cycle of your NI products for up to 20 years. You can use this preconfigured program as is or customize it to meet your specific application needs.

Reduce obsolescence risk and ensure long-term serviceability with the NI Long-Life Service Program.

Software Tools and Architecture

The test software architecture you select may be even more important than the software platforms you choose to build a sustainable test system.

COTS Software Tools

As mentioned, using COTS software tools can significantly increase the sustainability of test systems. COTS tools have wide user bases and are maintained by large, specialized software development teams. This means your software organization's maintenance effort is significantly reduced. Consider an older application built to work on Windows XP that needs to be ported to Windows 10 as a result of Department of Defense mandates. If the application development environment (ADE) used to develop the code does not support Windows 10, the test program must be completely reconstructed.

Companies that sell high-volume enterprise software tools can also offer tailored purchasing agreements and software subscriptions that meet specific billing needs or defined terms and conditions.

Test Software Toolchain and Interoperability

The ideal software package should minimize the effort to develop and expand the test program and, thus, streamline the productivity of software engineers while minimizing their sustainment effort over the lifetime of the test system.

To match the pace of technology acceleration, maximize engineering efficiency, and minimize software maintenance effort, test systems must be flexible and use ADEs that can withstand structural changes by working with multiple hardware and software platforms. Software developers may be forced to use multiple ADEs for different projects if the tools do not meet the needs or interoperate with the tools of each project. Imagine that your test program requires a new measurement performed by an instrument that is not in the test set. If your development software does not support this new hardware, then you may have to substantially change the application.

To incorporate these changes into the test program easily and quickly, you need scalable software. Examples of tools that improve the programming experience include easy-to-use application programming interfaces (APIs) that minimize the need to learn hardware caveats and ready-to-use example code that serves as a starting point for any application. Software tools should also simplify performing new analysis on data acquired from hardware by providing analysis functions or delivering interoperability with tools like MathWorks MATLAB® or Python, which offer complex data analysis.

Software tools should also include features that maximize the ability to reuse code. Some software tools do this by helping you create libraries or code repositories through interactive configuration utilities or by supporting drivers that use a standard communication protocol to help you program multiple instruments with the same API. NI instrument drivers are built to support multiple families and generations of instrumentation to maintain code compatibility over time.

Modular Software Architecture

Don't get locked into an inflexible test program by building a monolithic architecture; instead, plan ahead by building layers that perform separate test operations. In a monolithic architecture, the test program for the unit under test (UUT) includes code that manages test flow control, test execution, UUT stimulus, measurement analysis, limit checking, result logging, operator user interfaces, and instrument resource scheduling. This single source of functionality means that any new test requirements that arise because of an obsolescence event force you to revalidate the entire test system.

Instead, create a modular software architecture that has separate code bases for all critical test system functions. Test management software like TestStand handles common test program tasks such as test flow control, test execution, result logging, limit checking, operator user interfaces, and instrument resource scheduling. Test code should be responsible for tasks specific to the UUT like stimulus, measurement, and analysis functionality.

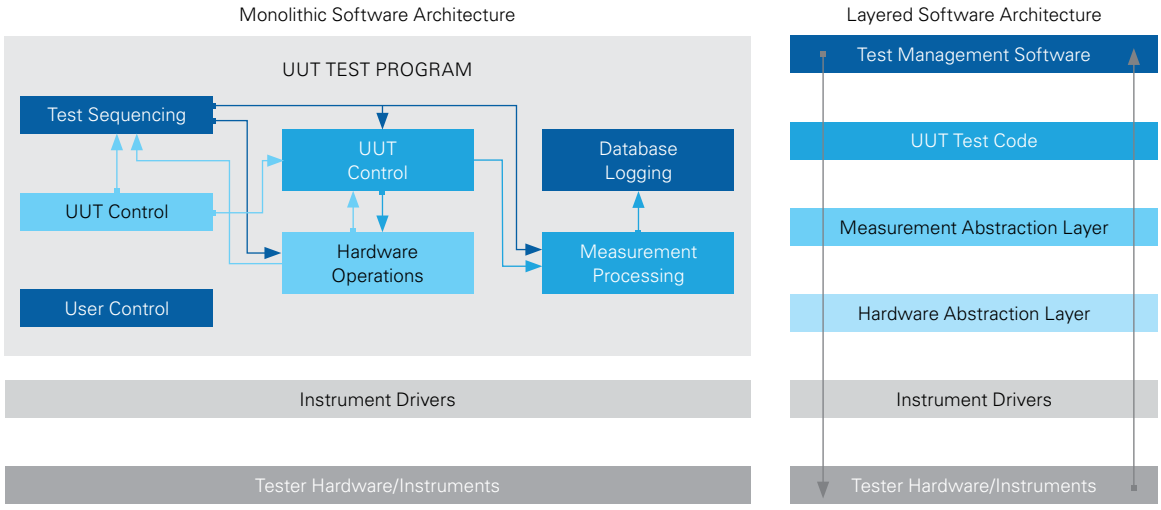


Figure 7. A single code base to handle all test program tasks seems like a good way to develop until it becomes inflated and difficult to change or repair. Using smaller, modular code bases for different tasks keeps a test system more extensible.

Functional Abstraction Layers

Perhaps the most significant software technique to protect a test system against inevitable hardware obsolescence events is using hardware abstraction layers (HALs) and measurement abstraction layers (MALs).

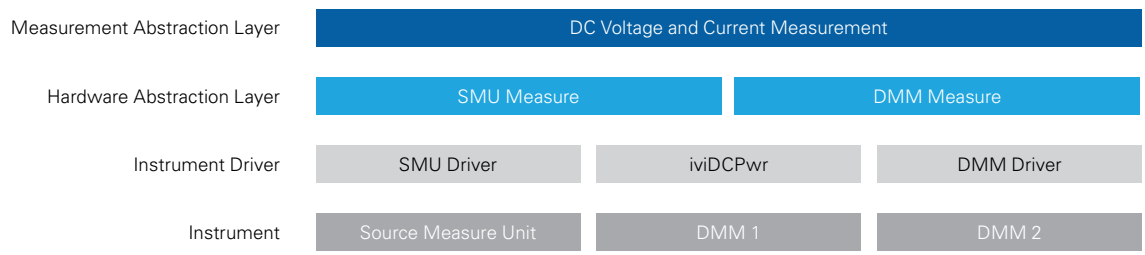


Figure 8. A MAL and HAL empower test engineers to choose the test result needed and allow the test system architect to maintain instrument driver and hardware interoperability.

Industry-standard instrument drivers like IVI can provide a quick and easy starting point for function abstraction, but they often fall short when you want to use specific features of new instruments that do not conform to standard driver function calls.

MALs help you develop high-level code that performs necessary functions without your defining specific instrument settings or communication. They also give the test system the ability to choose the correct and available resource to deliver a given test function. In some cases, a function in the MAL translates to a specific instrument, but some instrument functions overlap and could be used to complete tests in place of a busy or malfunctioning device. A good example of this is taking current measurements with a DMM. You can use a source measure unit (SMU) to take that measurement in many cases more effectively.

For a MAL to operate properly, you need an abstraction layer that handles instrument selection and communication. This layer is the HAL, which enables the code base to execute the function in the MAL from any specific instrument and device configuration in the system. Building these layers into your code gives you the flexibility to change instruments without altering measurement analysis code, the tester's user interface, or the overall test structure.

Learn the best practices and see the functional implementation NI test engineers used to develop an effective hardware abstraction layer (HAL) and measurement abstraction layer (MAL) architecture.

Technical Support and Training

A best-in-class vendor has on-demand technical support engineers to assist with any obstacles you may encounter or with getting started using hardware. NI offers on-demand technical support, classroom and online courses, and skills certifications programs. Certifications help you as a team leader measure the skills of developers on your team and give you a method to evaluate new engineers or contractors who may join your teams to help complete projects.

Learn more about NI training courses, professional certifications, and on-demand skill certifications in the NI badge program.

Next Steps

Discover strategies for handling reactive obsolescence events.

Explore the NI test engineering HAL and MAL in this 30-minute webinar.

Learn about NI's solution for electronics maintenance test.