WHITE PAPER

# Accelerating the Product Development Lifecycle with Faster Test: Navigating the V Diagram

CONTENTS

Electromechanical System Development Process ..........2

Model-in-the-Loop (MIL) Test ..........5

Software-in-the-Loop (SIL) Test ..........5

Processor-in-the-Loop (PIL) Test ..........5

Rapid Control Prototyping (RCP) ..........6

Physical Test ..........6

System Integration Test ..........6

Maintenance, Repair, and Obsolescence (MRO) Test ..........6

Field Test ..........7

Designing a Unified Test Architecture: Servicing Test Needs Across the Design Cycle ..........7

Designing a Unified Test Architecture: Integration and Interoperability ..........8

Designing a Unified Test Architecture: Data and Systems Management ..........8

Test Better with a Unified Test Architecture ..........8

Next Steps ..........8

## Electromechanical System Development Process

Electromechanical systems, such as the propulsion system on an aircraft, consist of software, controller(s), and mechanical components. As they continue to increase in complexity, these systems need more sophisticated test methods to achieve the test coverage required within project schedule and budget constraints. Also, these testers must be maintained over multidecade program lifetimes to fulfill maintenance-repair-obsolescence (MRO) service contracts and periodically modified to accommodate system upgrades.

Supporting increasingly complex devices under test (DUTs) and test systems over these long program life cycles is challenging, especially with limited resources. A flexible platform-based approach to test can help you develop a unified test architecture to address these challenges. Unifying on a single test architecture offers several benefits including reduced test development time and minimized replication effort for test teams working in different groups across the development cycle or across programs.

For maximum benefit, a unified test architecture for electromechanical systems should:

- Service test needs across the design cycle from early prototyping to software, electrical, and mechanical validation to system-level test rigs and systems integration labs ("iron birds"), to production test
- Support model-based control and simulation to test earlier in the development cycle and to cover an expanded test matrix of hard-to-replicate test cases and stress tests.
- Integrate a wide variety of I/O and third-party devices such as sensors, actuators, instruments, and software. This maximizes reuse and minimizes integration work. A configurable/expandable and distributed/synchronized I/O architecture must be available to accommodate the different I/O needs for test cases across the design process and for reuse across programs.
- Provide enterprise-level and IT-friendly data management and systems management frameworks to improve decision making, reduce repeat tests, reduce reporting time and effort, and increase asset utilization and uptime.

Electromechanical systems convert energy to mechanical work and vice versa. They feature control electronics (for example, an electronic control unit) or embedded controllers (for example, a line-replaceable unit) running purpose-built software that controls mechanics, actuation, and physical components using inputs from sensors and other systems. Electromechanical systems provide vehicle propulsion and serve a variety of other functions on aircraft, ground vehicles, and ships.

ELECTROMECHANICAL SYSTEMS IN VEHICLES

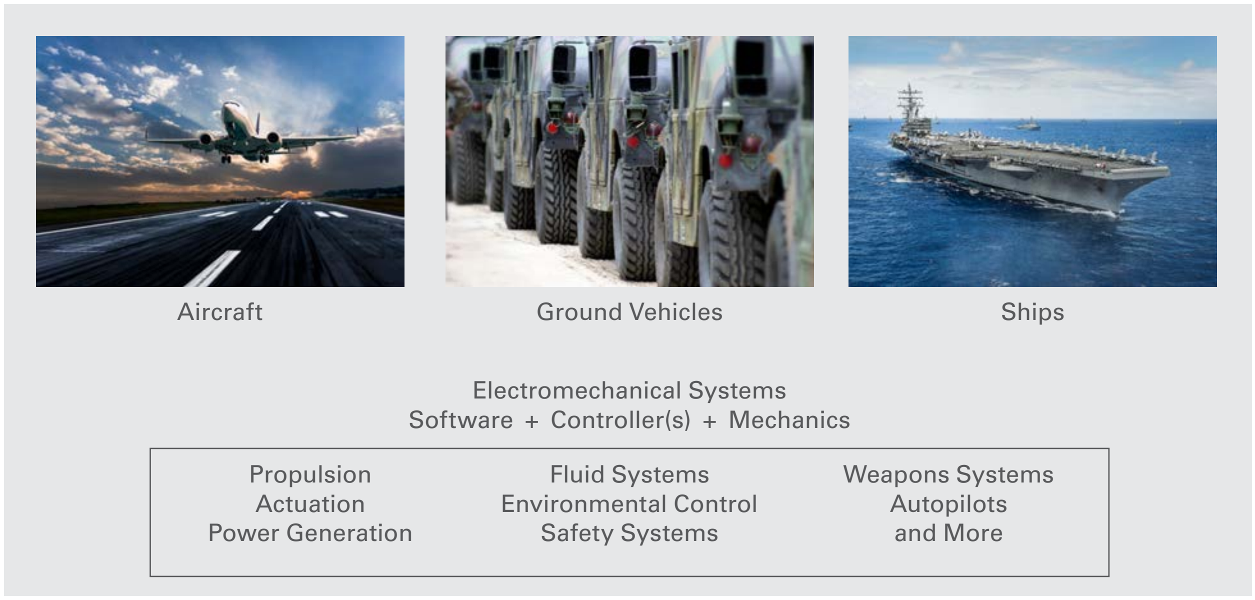

Figure 1. Example Vehicle Categories and the Common Types of Electromechanical Systems in Them

Though these systems' functions, methods of action, and design requirements can vary widely, the development and test of electromechanical vehicle systems follow the same general workflow. Systems engineers design the overall vehicle and the requirements that must be met by the vehicle's systems, subsystems, and components. Separate teams address these requirements by developing the appropriate control electronics, software, and mechanical components against specifications. These teams follow their own development processes (typically a common methodology such as Agile adapted to specific organizational needs) to progress through the design and validation steps for their specific pieces of the electromechanical vehicle system. Then, the system is iteratively built up, integrated, and tested in stages to produce the finished vehicle.

PRODUCT DEVELOPMENT PROCESS

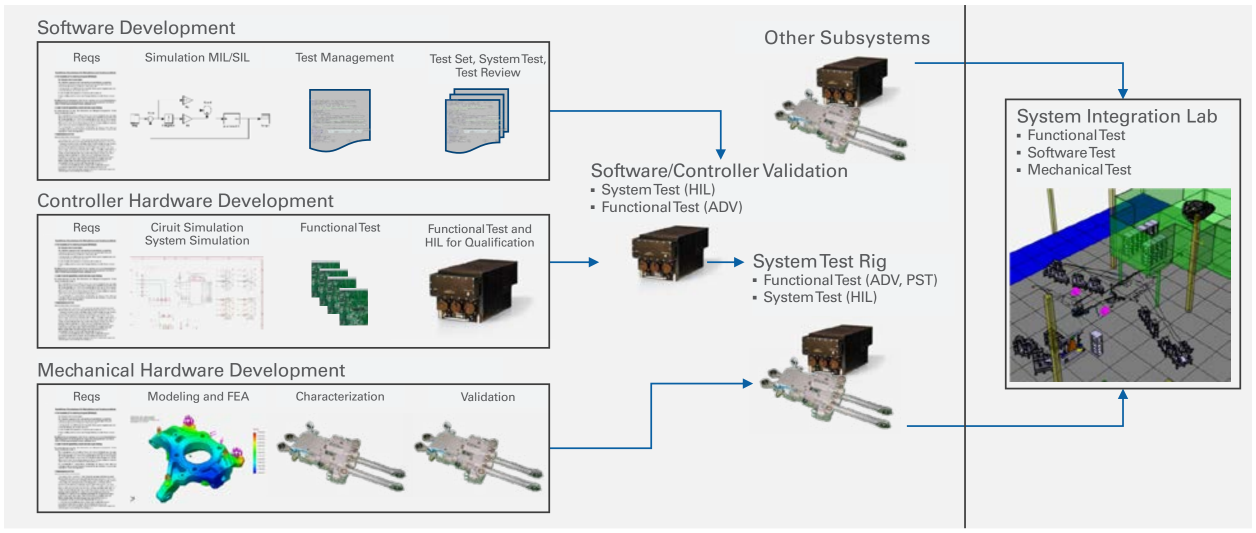

Figure 2. Generalized Product Development Process for Electromechanical Systems

The process of progressing from requirements through design and validation is represented in various ways, one of which is the V-model (Figure 3). Even with all the confusion surrounding the V-model and its many incarnations and interpretations, it is still a useful framework for examining the merits of a universal test architecture to address test needs across the design process.

V-MODEL

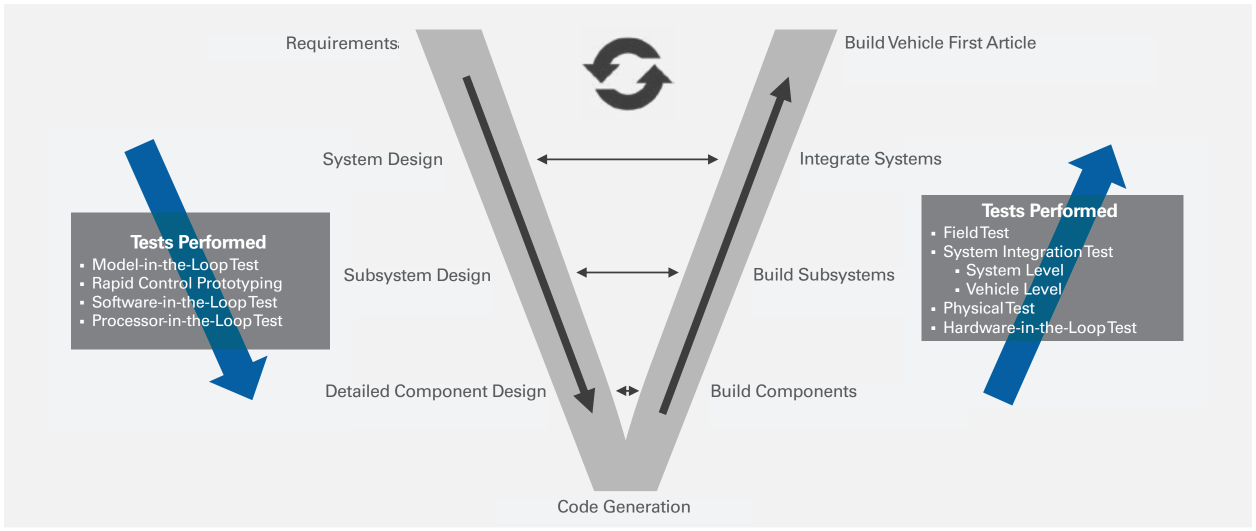

Figure 3. The V-Model Method of Representing the Development Process and Associated Test Types

In general the left side of the V represents the breakdown of high-level requirements to lower-level specifications through progressive design decisions. These decisions are made using an increasingly model-based design approach that encompasses a variety of computer aided engineering (CAE) tools depending on the type of system under development. For more discussion on this topic, see the substantial literature on digital transformation and digital twinning. At the bottom of the V, the systems have been broken down into their lowest base-level components, the design is ready to be translated into implementation, and the component prototypes are ready to be built (and code deployed to the prototypes that run software, which is why the bottom of the V is sometimes labeled "deploy"). The right side of the V shows integrating these various pieces together, verifying their operation against specifications, and validating performance against requirements at each integration step from base-level components up to the vehicle level.

Note: As the system, subsystem, and component details are provided, the development of software, electrical components, and mechanical components progresses in parallel as shown in Figure 2. Separate design and test teams with the requisite domain expertise usually own the development process.

Note: The terms "verification" and "validation" are used somewhat indiscriminately, and sometimes the umbrella term "V&V" is used. Generally, during verification, you ask, "Am I building the thing right?" and make sure your system operates as expected (with tolerance or range) compared to a set of specifications. But during validation, you ask, "Am I building the right thing?" You make sure your system, once complete, can perform the tasks it was designed for with acceptable performance as measured against a set of requirements.

The short descriptions and definitions for the following tests are roughly in the order that you encounter them in the product design process, which is highly iterative in practice. The ability to quickly and efficiently progress through the design V to iterate on design is a competitive advantage and a key competency of a best-in-class test organization. One critique of the V-model is that it represents a sequential process like the Waterfall design methodology.

## Model-in-the-Loop (MIL) Test

For MIL test, you model both the controller and the plant in software. You conduct this type of test early in the design process to test controller strategies and system behaviors in a software simulation environment.

## Software-in-the-Loop (SIL) Test

For SIL test, you model the plant, but you interface it to actual control code written and executed in your selected development language and compiled and executed on the development system, sometimes in a virtual machine.

## Processor-in-the-Loop (PIL) Test

PIL test is similar to SIL test, but the control code is written and compiled for the specific processor architecture and OS to be used in the deployed system. For instance, if you are running the code on a specific FPGA with specific settings, you compile the code for that specific scenario to ensure proper functionality on the actual processing architecture and to guarantee the availability of enough resources. This is a separate named test step because implementation of the deployed processing architecture and hardware can be a tricky and time-consuming process, especially when design optimizations in the electronic control unit (ECU) cause limitations and trade-offs on software functionality and features.

## Rapid Control Prototyping (RCP)

RCP is used for rapidly iterating on possible control schemes using mathematical models running on a real-time controller and/or an FPGA connected through real I/O to a real plant.

Hardware-in-the-Loop (HIL) Test

HIL is software testing on the actual embedded controller with the surrounding physical components and sensors simulated so the ECU is performing its real electrical I/O with signal conditioning, real load levels, and the ability to do fault insertion

## Physical Test

Physical test applications use transducer-based measurements (such as temperature, pressure, stress/strain, sound, acceleration, displacement, and so on) to test the physical characteristics of the component or system under test. Application examples include noise, vibration, and harshness (NVH) test, which involves taking sound and vibration measurements from microphones and accelerometers. Another example is durability test/life-cycle test (highly accelerated life test, or HALT, and highly accelerated stress screen, or HASS), which focuses on determining how the DUT will behave under a variety of different anticipated operating conditions.

## System Integration Test

The whole right side of the V-model emphasizes successive levels of integration test. Integration typically involves using two key types of systems:

1. Test rigs are used to test the integration of various components into a single (sub)system for system-level test. Test rigs are well suited to handle a wide range of possible tests because they bring together the various base-level components into a functioning system. Having a functioning system facilitates test setup and execution.

2. The system integration lab or "iron bird," which is the integration of multiple subsystems for testing at the vehicle level, approximates the actual vehicle layout and system connections. It is used for system-of-systems test and intersystem communications/interactions/boundary conditions test. Some test cases can only be covered by having this magnitude out of test infrastructure. This is the last level of testing before a prototype vehicle(s) is built for actual field/flight testing/trials.

## Maintenance, Repair, and Obsolescence (MRO) Test

MRO or depot test provides for the services, repairs, and upgrades vehicle systems need over their long, often multidecade and multigenerational lifespan. Sometimes the same equipment, or a modified version of it, is used for system test rigs. A challenge with MRO test is efficiently maintaining test capability over the life of a program throughout tester hardware and software obsolescence issues, personnel turnover, and changing requirements due to periodic system upgrades.

## Field Test

Vehicles used for field test are usually heavily instrumented to provide as much data as possible about the operation of the vehicle during this expensive type of test. Key considerations are weight/size, ability to power the test equipment, and data storage and retrieval. Though time intensive and costly, field test is still a key part of the product development process because models are never perfect, and system interactions are complex and can result in unanticipated behavior that was not covered in the preceding test procedures. Field test determines operational readiness.

## Designing a Unified Test Architecture: Servicing Test Needs Across the Design Cycle

Now that you're familiar with the development process and associated types of test, you can see the potential advantage of a unified test architecture. The ability to service test needs across the design cycle, from early prototyping to software, electrical, and mechanical validation to system-level test cells and system integration labs, with one test platform allows for faster test development and more efficient resource utilization. Both equipment and people are more interoperable and fungible across the design V and across programs. This ability to more quickly and easily progress through the V-model with respect to test capabilities and iterations is extremely valuable.

A unified test architecture offers benefits similar to object-oriented programming models. If you know you need to develop generally the same type of systems using similar methods, you can invest in core building blocks that can be used across projects and customized for the unique specifications of each project.

This platform-based approach to test is more robust because with a flexible, extensible platform underpinning your test architecture, you can be confident that changing requirements and unanticipated future demands on the system will not "break you" in terms of system functionality and test capability. Or, to put it another way, with this approach, you can prepare for the unknown but anticipate for future requirements by not explicitly designing out functionality. This is much better than purpose-built fixed-function systems for which you must explicitly design in the flexibility and extensibility you think you may need in the future. These fixed-function systems force you to make time and cost versus capabilities trade-offs.

Designing a Unified Test Architecture: Supporting Model-Based Control and Simulation

You need to "front load" as much test as possible as early as possible in the development process to iterate faster and ensure cheaper and safer tests. You can do this by using model-based control and simulation to adequately represent stimuli and real-world conditions in the lab. This helps you move analysis previously possible only with field test into the system integration lab or system-level test rigs. Techniques such as recording real-world stimuli in the field and "playing them back" on the system through model-controlled actuation are improving this type of test.

You can also decouple test requirements from other teams' schedules by simulating their components. But to ensure sufficient test result fidelity, you must spend time validating the accuracy and efficacy of the models and methods used to simulate components.

Another benefit is the ability to better test hard-to-replicate, dangerous, and extreme test cases. This increased test coverage results in better confidence in the design.

## Designing a Unified Test Architecture: Integration and Interoperability

Sometimes you don't have a lot of choice in the type or brand of sensor, actuator, instrument, or software you can use for a particular function. Making all the disparate pieces in a system work together is often a significant portion of a test system's design cost, especially when dealing with aging system components in a legacy tester retrofit or redesign. The ability to leverage a platform with a wide variety of I/O and third-party device capabilities built in can improve efficiency by maximizing reuse and minimizing integration work. A configurable/expandable and distributed/synchronized I/O architecture accommodates the different I/O needs for test cases throughout the design process and promotes reuse across programs.

## Designing a Unified Test Architecture: Data and Systems Management

More measurements at higher rates are creating more data—much more data—in a variety of formats. And more clients must access this data across the various types of test in the design cycle to meet more demands for reports. Ensuring data is usable—and that it's actually used—is already challenging. You need to be able to effectively locate and load data, interactively visualize and analyze that data, and save time by automating reporting. A unified test architecture must provide enterprise-level and IT-friendly data management and systems management frameworks to improve decision making, reduce repeat tests, decrease reporting time and effort, and increase asset utilization and uptime by making data an asset, not a liability.

Increasingly, globally distributed development and test teams present challenges in effectively managing testers deployed across those sites and correlating the data generated throughout them. The effective management of distributed test systems and data can generate big savings when it comes to improving operational insights, decreasing downtime, and increasing confidence in generated data.

## Test Better with a Unified Test Architecture

Investing in a unified test architecture based on a software-defined test platform is a best-in-class approach for teams designing and testing advanced electromechanical vehicle systems. This approach enables faster test development, better test coverage, more efficient operation, a nimbler and more capable team, lower capital expenditure, and better long-term test system uptime and maintainability compared to full in-house custom development or full turnkey outsourcing.

## Next Steps

- View this webinar on future-proofing test system design using a platform-based approach.
- Learn more about platform-based systems configurations for specific types of test.