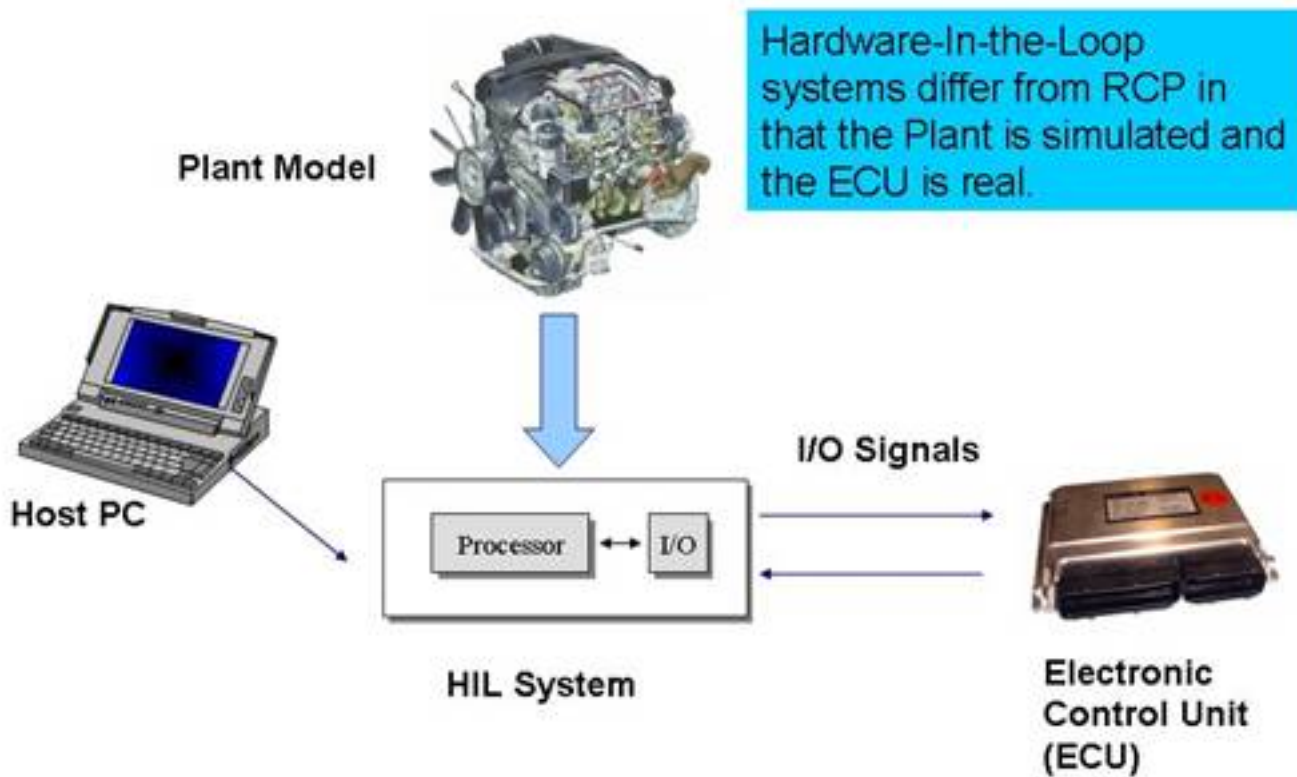


Hardware-In-the-Loop (HIL)

Hardware-In-the-Loop is a form of real-time simulation. Hardware-In-the-Loop differs from pure real-time simulation by the addition of a “real” component in the loop. This component may be an electronic control unit (ECU for automotive, FADEC for Aerospace) or a real engine. The current industry definition of a Hardware-In-the-Loop system is shown in Figure 1. It shows that the plant is simulated and the ECU is real. The purpose of a Hardware-In-the-Loop system is to provide all of the electrical stimuli needed to fully exercise the ECU. In effect, “fooling” the ECU into thinking that it is indeed connected to a real plant.

Hardware-In-the-Loop (HIL) Definition



©2005 PrecisionMBA, LLC

Figure 1. Hardware-In-the-Loop Definition: Plant is simulated and ECU is real.

The benefits of a Hardware-In-the-Loop system as defined above are manifold. This topology was devised to manage the ever-increasing complexity of ECUs. Open loop or “stimulus boxes” were no longer suited to testing ECUs. Why? First off, the ECUs

were closing dynamic controls loops to manage, say, fuel mixture. Open loop stimulus-boxes cannot test dynamic closed loops. Second, automated methods were needed to test and verify all the features of the ECU. Hardware-In-the-Loop systems typically have the ability to automatically run through tests automatically. We will talk about this more later.

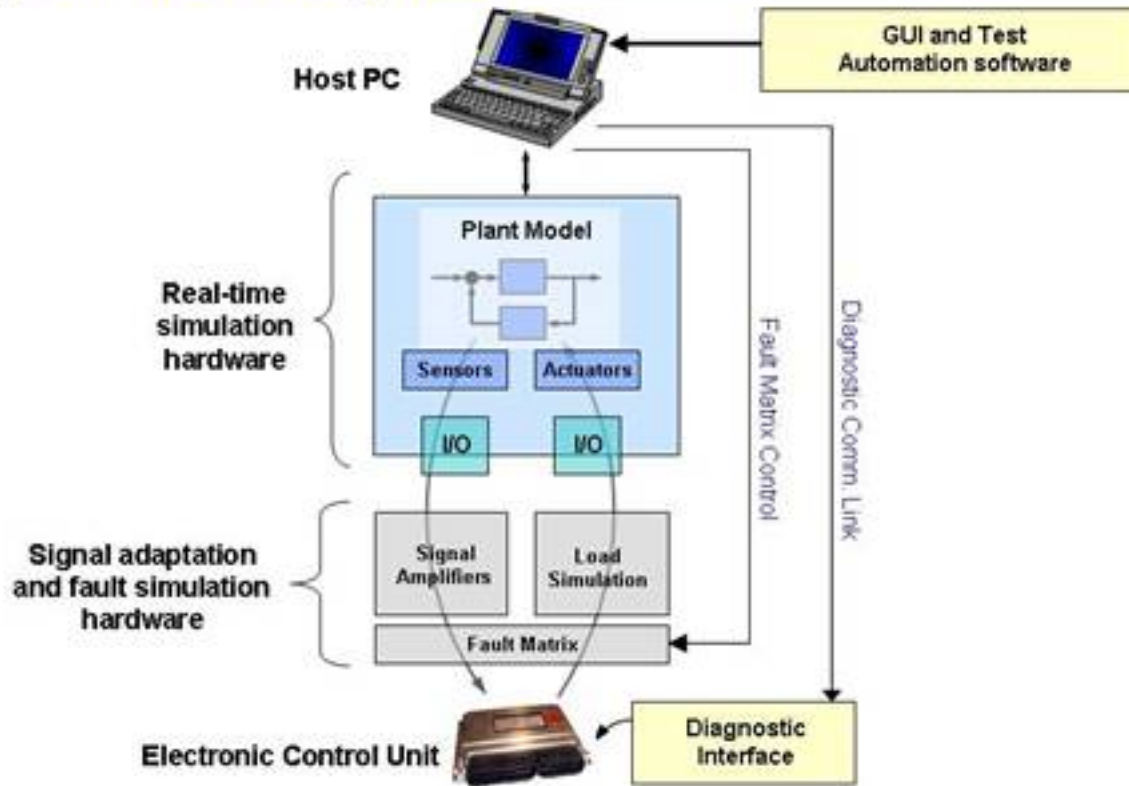
Another benefit of Hardware-In-the-Loop is that testing can be done without damaging equipment or endangering lives. For instance, potentially damaging conditions in an engine, such as over-temperature, can be simulated to test if the ECU can detect and report it. Another instance would be an anti-lock braking (ABS) simulation at performance extremes. If simulated, the performance of the ABS system can be evaluated without risk to the vehicle or operator.

If asked to divide the usage scenarios of Hardware-In-the-Loop systems, I would venture that 75% of systems are dedicated to testing the diagnostic code of the ECU. For example, fault simulation and detection. The other 25% of Hardware-In-the-Loop systems are dedicated to development work, i.e. the engineer is developing new ECU features against the simulated plant before taking them to the vehicle. This is somewhat counter-intuitive, but it does reflect the priorities of the Original Equipment Manufacturer (OEM). They need to be in compliance with government regulations such as On-Board Diagnostics (OBD). Fast fault detection can also prevent more serious warranty repair.

The typical Hardware-In-the-Loop system is comprised of the following components (see Figure 2):

1. A math model of the plant (i.e. engine or vehicle model).
2. Sensor models
3. A real-time target computer(s) with I/O.
4. Real or simulated loads
5. Fault insertion relay matrix
6. A host PC with communications link to target computer and diagnostic link to ECU.
7. A Graphical User Interface (GUI) application to download and control the real-time process.
8. A test automation application to automate all aspects of the test.

Typical HIL Test System



©2005 PrecisionMBA, LLC

Figure 2: Typical Hardware-In-the-Loop system components.

As you can see, the Hardware-In-the-Loop system is much more complex than an RCP system. Let's talk a little about all the components to give you a feel for what the system is.

The Model: The plant model is whatever the ECU expects; an engine model, vehicle model, airplane model, etc. The typical question is "where do I get the model, and how good does it have to be?" You can buy off-the-shelf models or make your own. Off-the-shelf models are available for spark and diesel engines, cars, and trucks. They are typically Simulink™ models designed to be run in real-time. You can also design your own plant models or have a specialist do it for you. How good do they have to be? That depends on your application. What I hear a lot is "necessary and sufficient." In other words the model fidelity must match its intended use. The typical use is for testing the diagnostic capability of the ECU. This would put the least requirements on the model. If you intend to develop ECU control capability against the model, it may need to have higher fidelity. For instance, if you are developing a new multi-injection fueling strategy for your diesel engine, your model should be

capable of accumulating and reacting to all of the fuel pulses.

Sensor Models: This aspect of the Hardware-In-the-Loop system is often overshadowed by the plant model, but it is nonetheless very important. The sensor outputs from a plant model are perfect - they are, after all, just variables. Unfortunately, real-world sensors are not perfect. Their greatest imperfection is their non-linearity. ECUs compensate for sensor non-linearity, thus, the imperfect sensor response must be modeled. The typical sensor model is realized as a look-up-table since the response is typically fast compared to the time-constants of the plant. If the dynamic response of the sensor is slow it can be modeled dynamically. Some Hardware-In-the-Loop systems will dedicate an entire processor to the sensor models, thus decoupling them from the plant model.

Real-time targets and I/O: The majority of Hardware-In-the-Loop systems uses embedded computers to run the models in real-time. The reason for this is to decouple the real-time computing of the Hardware-In-the-Loop system from the host PC. The PC with a MS-Windows® operating system is not real-time. There are some Hardware-In-the-Loop vendors that do use PC hardware for the run-time platform and they accomplish this by running a PC real-time operating system (RTOS) like QNX®. The embedded computers, a.k.a. "targets" communicate with each other and system I/O via a data bus. This bus can be VME, PCI, PXI, or proprietary. Their typical physical footprint is a card cage with processor and I/O boards connected with a passive backplane. The I/O boards are typically "instrument grade" in that they are not able to source or sink current. For digital I/O TTL is typical. For analog I/O, operational amplifiers are used.

Loads: The I/O provided by the Hardware-In-the-Loop vendor is instrument grade and some signals will need to be amplified and/conditioned to allow connection to the ECU. Loads in an Hardware-In-the-Loop system can be real or simulated. An example of a real load would be a known-good fuel injector. A simulated load would be an Inductive/Resistance (LR) network designed to simulate the frequency response of the injector and be able to sink and dissipate delivered current. The loads and signal conditioning of a Hardware-In-the-Loop system are inherently custom and match the requirements of the ECU under test. Hardware-In-the-Loop vendors anticipate this custom adaptation by designing load and signal conditioning cards that can be easily customized by component selection, i.e. stuffing different inductors, capacitors, or resistors.

Fault Insertion: The majority of Hardware-In-the-Loop testing time is used to test the fault detection capability of the ECU. Modern automotive ECUs dedicate half of their

memory to this task. Key to testing is fault insertion such as wire breaks, shorts to power, shorts to ground, or sensor/actuator failures. Simulating these events is a relay matrix between the Hardware-In-the-Loop and the ECU (see Figure 2). The relay matrix is controlled by the host PC as it steps through manual or automated tests. The link between the relay matrix and the host PC will vary by manufacturer, although some use Controller Area Network (CAN) since it is needed by the Hardware-In-the-Loop system anyway.

Host PC: A host PC is used to provide a GUI to the user, to run test automation applications, to control Hardware-In-the-Loop system components such as fault insertion switching, and to provide a diagnostic link to the ECU, to develop/change models and tests, and to collect, store, and report test results. The link between the host and the real-time system is typically Ethernet. Other methods such as proprietary high-speed serial or parallel bus are also used.

GUI: The GUI application runs on the host PC. This application allows control of the real time process such as download, start, stop, variable observation, and data collection.

Test automation application: This application may be built-in to the GUI or be a separate application that “sits on top” of the GUI. This application provides the user with the ability to automate tests. It accomplished this task by providing the user with a method of authoring a test sequence. The method can be a test script in a language such as Visual Basic® or Python® or it may be symbolic like flowcharts. There is currently no standard language for Hardware-In-the-Loop test automation.

Hardware-In-the-Loop systems are complex, expensive, and require a corporate commitment to be successful. Some full-vehicle multi-ECU Hardware-In-the-Loop systems can sell for over a million dollars. The benefits and payback period are not well documented but those that adopt the technology soon find it invaluable for testing and debugging ECUs. The automated test capability makes it possible to run a test-suite on every ECU software change saving countless man-hours over manual methods.

Hardware-In-the-Loop is currently THE method for managing ECU software testing.

If you are considering Hardware-In-the-Loop we can help you with the process. Please see our [vendor selection services](#) or [contact](#) us.

[Home](#)

©2005 PrecisionMBA, LLC