Advanced Techniques for Simulating ECU C-code on the PC

Vivek Jaikamal
ETAS Inc.

Thomas Zurawka
SYSTECS Informationssysteme GmbH

ABSTRACT

Over the last two decades, adoption of model-based techniques for the development of ECU software has resulted in major gains in productivity across the automotive industry.   However, the fact remains that the majority of the ECU software today is still hand-written using the "C" programming language. Further, the need to shorten the development time, reduce costs and increase the quality of the ECU software has driven companies to adopt virtual (PC-based) simulation techniques rather than rely on expensive in-vehicle and dynamometer set-ups. This has lead to a situation where the two development philosophies (models and hand-written code) need to be properly integrated in order to fully capitalize on the advantages of PC-based techniques. For the complete ECU system to be simulated, typically, automatically generated C-code from other tools must be integrated as well. Since current tools do not support the integration of hand-written or automatically generated C-code very well, virtual PC-based simulations (e.g. software-in-the-loop) are very time consuming activities and are, therefore, not broadly introduced in the industry. INTECRIO (from ETAS) and INCODIO (from SYSTECS) offer a powerful, commercial platform for PC-based simulation of embedded software, eliminating the need for proprietary solutions.

INTRODUCTION

Software development for automotive ECUs follows a typical development process that can be described using the traditional V-cycle (Figure 1). The steps in this development process have already been well described and documented [1].

Hardware-in-the-loop (HiL) is a well established methodology for testing of automotive software under simulated conditions. However, it requires the software to run on an electronic control unit (ECU), and the vehicle components (e.g. sensors and loads) to be either available physically or simulated accurately. It is a powerful method to test software repeatedly, automatically and without risk over a wide range of operating conditions of a vehicle. However, the availability of the physical hardware (e.g. ECU or loads) and the overall cost of the HiL system are, at times, impediments to the widespread deployment of such systems.

Over the last few years, the Software-in-the-Loop (SiL) methodology has emerged as an alternate approach to embedded software testing. Rapid advances in plant modeling techniques and automatic code generation combined with the availability of abundant computing power in modern desktop PCs has made SiL particularly attractive today. A SiL system requires no physical hardware and can reduce the time and costs of new software development considerably. However, the big benefit of SiL is in the PC simulation of hand-written

code that automotive companies have created and deployed in vehicles over decades. This software has been refined and tested in millions of vehicles and, as a result, its re-use is very attractive for quality and reliability purposes. New ECU functions, however, are more often created using model-based design (MBD) techniques coupled with automatic code generation. Even so, most new embedded software applications are composed of 80-90% pre-existing (or legacy) code with minor modifications. Therefore, a SiL environment must be capable of integrating legacy code with models or automatically generated code. On the other hand, it should also be capable of interfacing with plant models, much of them domain specific, from various sources and suppliers.
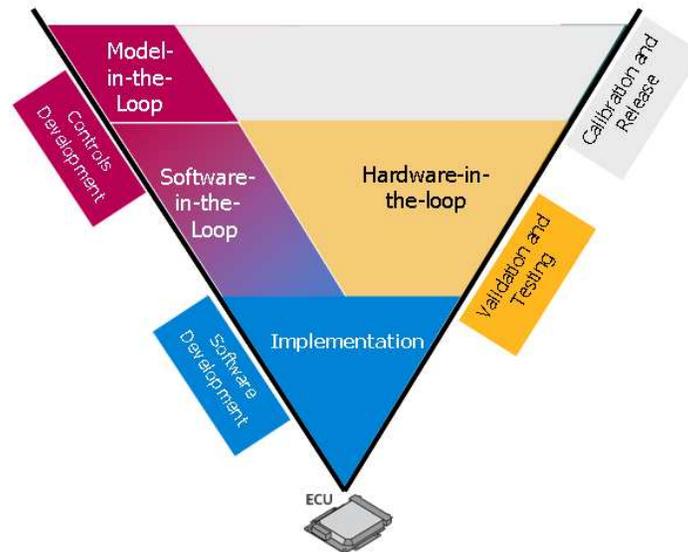

Figure 1: ECU software development process (V-cycle)

MBD tools (such as ASCET [9], Simulink [10]) typically offer a built-in mechanism to interface with external C-Code which is convenient but usually suffers from one or more of the following deficiencies, mainly because the C-code is treated simply as a "black-box" by these tools.

- The C-code has to be wrapped by interface code that defines the input and output variables as well as internal parameters of the code that can be accessed during run-time. This interface is usually fixed and only provides limited access to the variables inside the code.
- Missing references have to be resolved manually by the user. This is a huge disadvantage since a SiL system is usually built-up incrementally.
- Calibration of curves and maps in the C-code is not supported.
- C-code functions cannot be scheduled to execute under strict time or event based triggers from a virtual OS. This is necessary for the SiL system in order to accurately represent the run-time conditions on an ECU as closely as possible.
- Information (signal data types, conversion formulas etc.) contained in available artifacts (e.g. ASAP2 [11] description files) cannot be used.
- Automatically constructing a system by connecting signals (variables) from a large number of C-code files together is not easy. As a result, engineers spend a disproportionate amount of time constructing a SiL system rather than in verification and validation tasks.

In order to overcome these challenges, many companies have invested in creating their own custom environments for SiL systems. These environments usually require a large number of scripts and GUIs to be created and maintained, making them cumbersome to use and complex to upgrade to meet changing

requirements. In this paper, we present a commercial solution developed together by ETAS and SYSTECS that offers an innovative SiL platform for the virtual development, testing and calibration of ECU software while overcoming the limitations of the traditional approaches prevalent today.

THE ELEMENTS OF A SIL SYSTEM

Code and Models

As already discussed, a SiL system is made up primarily of an ECU software module (c-code) interacting with a corresponding relevant plant model on the desktop PC. For example, if an engineer wants to develop and test an engine idle control software module, he or she also needs a relevant plant model which, in this case, could be a simple look-up table (engine speed based on throttle and load) as shown in Figure 2.
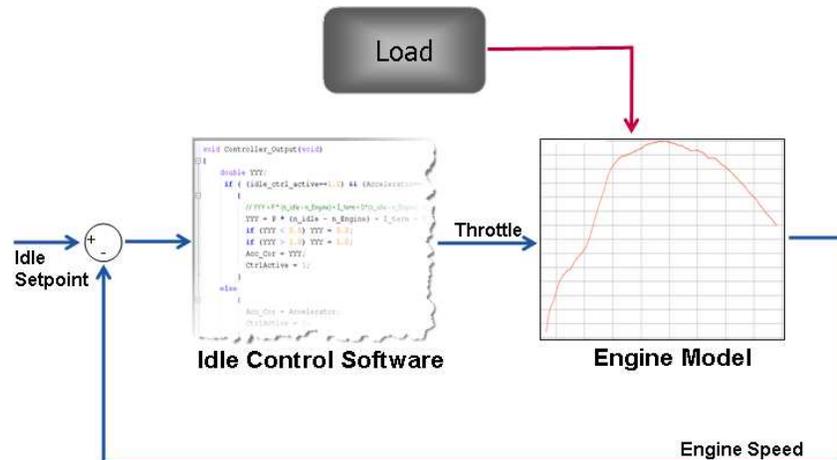

Figure 2: A simple idle control SiL system

A simple SiL system, such as the one above, can be used to quickly test new code in a single module, minor modifications in existing code or to calibrate the code to meet design criteria. This is traditionally known as module functional testing, or unit testing.

One can easily extrapolate the simple picture to include multiple software modules and the corresponding relevant plant models. New code modules (features) are typically added incrementally to satisfy the interactions needed by previous modules, until a full vehicle domain (e.g. powertrain, chassis, body etc.) is created. Similarly, additional plant model components may be added on an as-needed basis. This results in a system-level SiL simulation (Figure 3), which takes some effort to set up, but is extremely useful to conduct a full inquiry into the behavior of the ECU code. In order to facilitate the connections between the ECU software signals and the plant model, a software module known as the "virtual signal bus" is introduced. This module can absorb all the signal translations necessary in the absence of hardware. Usually, cross-domain signals (e.g. engine speed signal needed for a transmission controller) are provided either by a simulation of the network traffic on the communication bus (e.g. Controller Area Network or CAN), or directly via the simulation of those components.
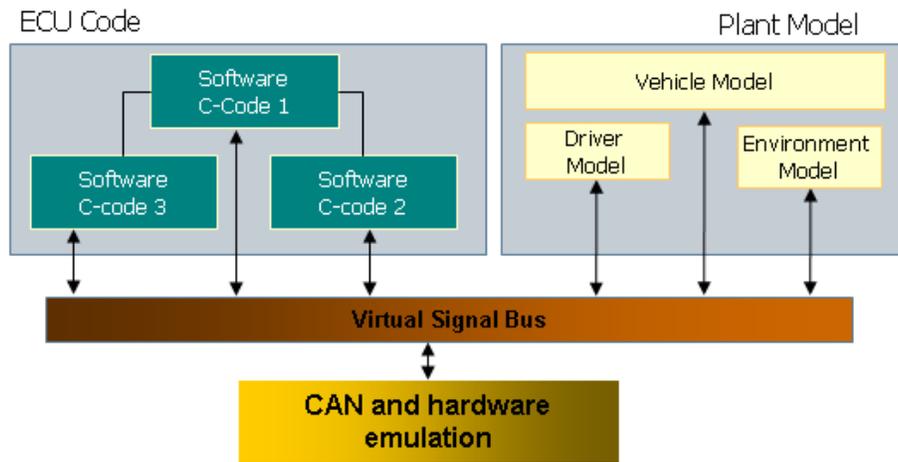
Figure 3: A system-level SiL setup

The SiL Integration Environment

In addition to the model elements, one key element for a successful SiL set up is the integration environment (Figure 4). The integration environment provides the necessary glue to manage the SiL system - e.g. to connect the models and code together, enable time or event based scheduling of the code under a virtual OS and to provide a run-time control, calibration and measurement interface to the user. Since ECU software components typically run at different task rates, the integration environment must support buffered data communication in order to maintain consistency. Finally, the integration environment allows the creation of logical software architectures similar to those implemented on an actual ECU (e.g. using functions and modules).
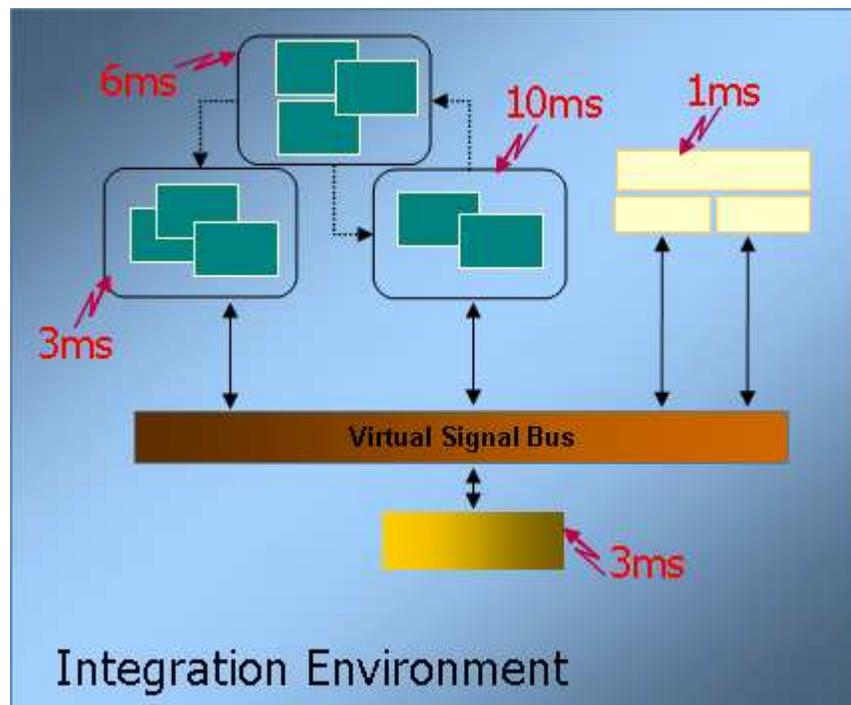

Figure 4: The SiL Integration Environment

In some cases, it may not be possible to integrate the plant models and ECU code into a single system. This is usually the case when the plant models are built in their own proprietary platform and it is not possible to convert the models to match the needs of the SiL integration environment. In such cases, the integration environment has to support the co-simulation mode, where the ECU code (discrete sub-system) and the plant model (continuous sub-system) are executed in their own simulation environments yet interconnected via the integration environment (Figure 5). Co-simulation is usually realized by taking advantage of the interfaces of the plant modeling environment (e.g. COM-API or proprietary). The integration environment acts as a master controller, issuing commands to the plant model environment to advance one simulation step at a time, reporting the results back to the ECU software components which are themselves executed at their native task rates in the virtual OS.
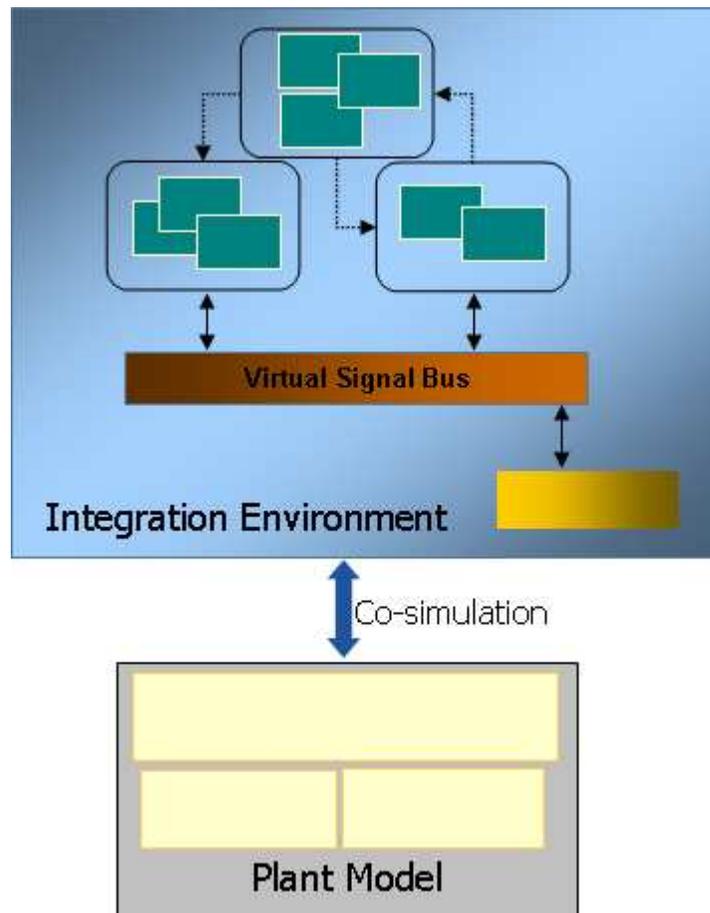

Figure 5: A Co-simulation framework for SiL

It is also important that the integration environment have a published API that advanced users may use to automate the SiL testing. An open and flexible integration environment is, therefore, critical to the setup, operation and management of the SiL system.

INCODIO – PREPARING C-CODE FOR SIL

The first tool offered in this paper is INCODIO, created by SYSTECS [2]. The goal of INCODIO is to accelerate the migration of production C-code to a PC environment where it can be combined with other model-based algorithms, plant models and automatically generated code to create a variety of SiL systems to fit development needs.

It should be noted that, in some cases, the original ECU software is not very well separated into the functional (i.e. pure control algorithm based and hardware independent) part and the non-functional (i.e. hardware dependent) part. The non-functional parts of the software may consist of special code necessary for optimum utilization of target resources, such as:

- Microcontroller operations (e.g. special bit-wise operations)
- Math libraries (e.g. interpolation routines and fixed-point arithmetic functions in assembly language)
- ECU architecture (e.g. special memory layouts for calibration variables)
- Compiler directives (e.g. "pragma" instructions, #if and #elif pre-processor commands)
- Real-time operating system (RTOS) scheduling.
- CAN bus interactions and hardware I/O (e.g. analog and digital signals)

Nevertheless, the basic requirement imposed on the SiL system by most companies is to execute the C-code from the production environment "as-is" on the PC. This imposes some challenges on how the non-functional code can be handled when preparing the ECU code for SiL. There is no general solution to this challenge at this time, mainly because it depends highly upon the structure of the code itself. In some cases it is as simple as abstracting the hardware interfaces into a new software module which is then simulated in sync with the rest of the software on the PC. For CAN and hardware I/O signals, proper emulation of the relevant signals on the PC is needed. Emulation of the RTOS on the PC is necessary in order to maintain the execution order and timing of the code. Most companies now follow a strict modular approach to embedded software development which requires a clear separation of core functional components from the non-functional components. Such a paradigm offers a huge advantage in terms of reducing the barriers for SiL system setup.

INCODIO is designed to maximize the re-use of pre-existing information about the code's structure and layout available in a production ECU software build environment. The use of this information, such as that listed below, relieves the engineer from developing proprietary tools (e.g. scripts that parse the code to extract information) and accelerates the set up of a SiL system considerably.

- Source (.c) and Header (.h) files from both model-based or c-based development tools
- Description files for the interfaces of each software module (e.g. definition of inputs and outputs according to the AUTOSAR[12], MSR or ASAM-MDX [11] standards)
- Description files for capturing the real-time behavior of the code (e.g. definition of processes and timing parameters according to the OSEK-VDX[13], MSR or AUTOSAR standards)
- Description files for scalar parameters, curves and maps.
- Description files used for measurement and calibration of internal code values (e.g. ASAP2 files based on the ASAM-MCD standard)

The main window of INCODIO allows you to create a library of software components. Each software component is composed of multiple source and header files along with the associated description files (Figure 6). Once the files are added, INCODIO automatically parses all the information to create a view of the software component it will create. For example, the ASAP2 file entries are used to determine which software identifiers are calibrations (i.e. characteristics) and which are measurements. This information is automatically added to the "Usage" field. In case certain identifiers are missing from the ASAP2 file, their usage can be entered manually. The input and output direction of code variables can also be specified in this view. Formulas for conversion of software data types from fixed-point to physical are also obtained from the ASAP2 file. This is necessary, since this software component will interface with plant models and other control system models that may use or provide the data in a physical format.
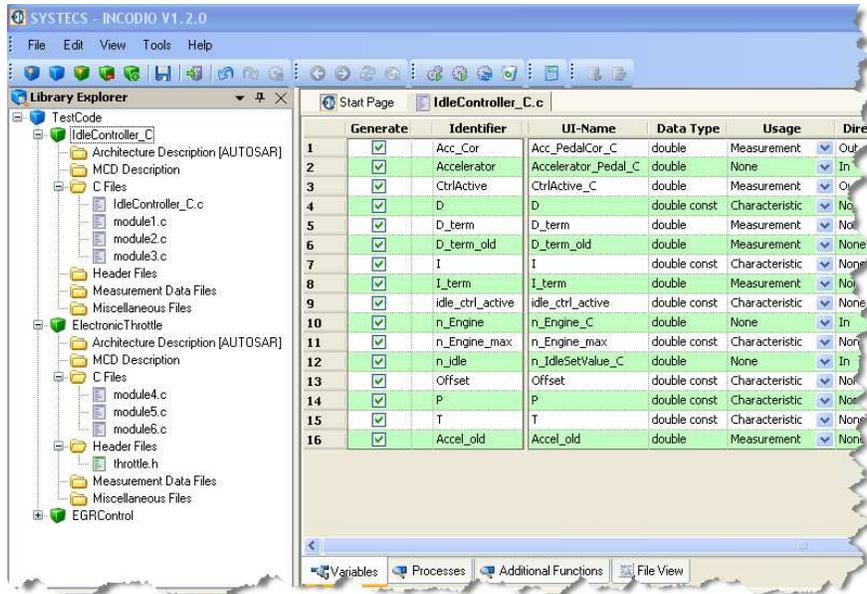
Figure 6: Creating Software Components in INCODIO

INCODIO also detects functions in the C-code that can be scheduled directly from the virtual OS (Figure 7). These functions typically do not have any input arguments or return values. The user is then given the option to specify how these functions will be called (e.g. in a timer, software, interrupt or initialization task). The period and offset of timer tasks can be specified here as well.
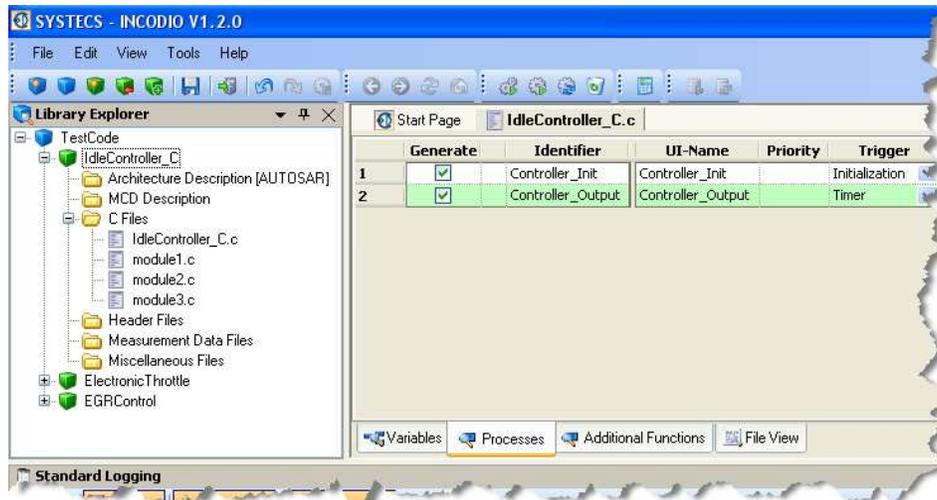

Figure 7: Process identification in INCODIO

Once all the information has been added, INCODIO automatically generates the final software component (Figure 8) and the associated description files, including:

- input and output interfaces for connecting to other software modules and plant models,
- processes for scheduling C-code functions on a virtual OS (OIL file), and

- measurement and calibration handles for interacting with the code on the PC (ASAP2 file)
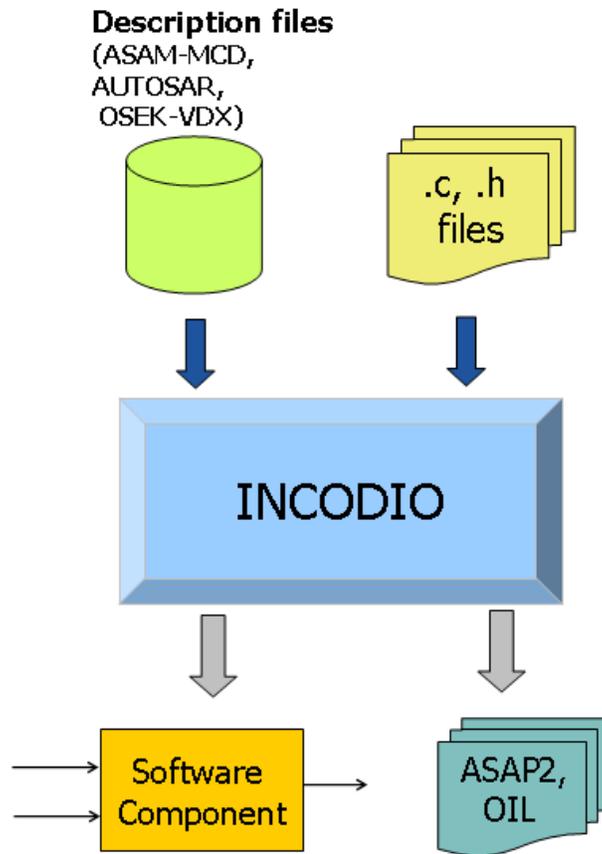

Figure 8: INCODIO high-level architecture

The source files remain untouched during this process. In addition, INCODIO can verify the software component by compiling the sources to check for syntax errors or missing/duplicate variable definitions. A PC compiler is included for this purpose. INCODIO also offers the option to use your own compiler, extend the parser via user-defined macros and specify include directories for additional files. If needed, INCODIO can create global definitions for missing symbol references.

INTECRIO – INTEGRATING AND SIMULATING C-CODE

The software component created by INCODIO is designed to be read directly by the second tool, INTECRIO, created by ETAS [9]. INTECRIO can read several C-code components created by INCODIO, model-based code from Simulink using Real Time Workshop (RTW) or Embedded Coder and code from ASCET models (Figure 9).
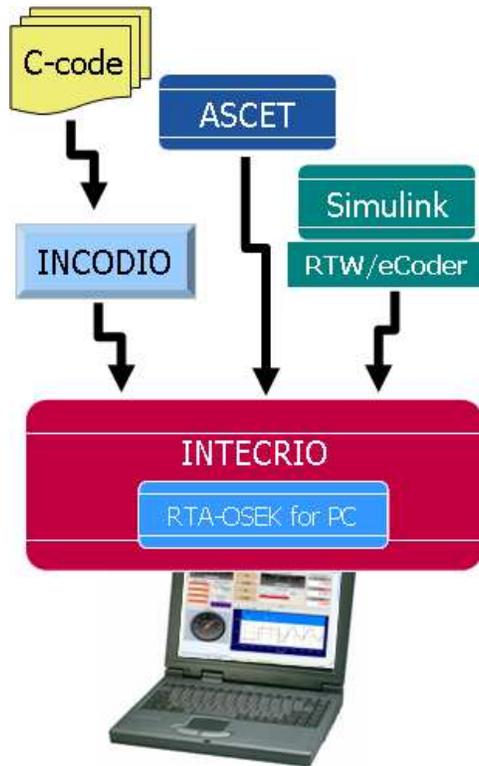
Figure 9: INTECRIO as a SiL Integration Environment

INTECRIO also includes a virtual OSEK-based OS (RTA-OSEK from ETAS [9]) that runs under Windows, allowing full ECU-like scheduling (with preemption, task delays etc.) for each software component. Individual C-code functions may also be scheduled using the process handles created by INCODIO. The OS in INTECRIO also allows users to create their own event driven tasks as well as manage the order of execution of various processes within a task. In this way, INTECRIO acts as the integration environment for SiL systems, as discussed earlier. Figure 10 shows an INTECRIO SiL system for the engine idle controller example, with a software system created from components imported from INCODIO and Simulink, and a simple vehicle model (plant) created in Simulink.

In the INTECRIO workspace, the ECU software architecture can be emulated in logical hierarchies (Figure 11). This feature allows engineers to create a repository of model and code components that they may logically combine to create systems as needed in their projects. As an example, the lowest level software functions can be imported as "Modules" - e.g. INCODIO software components or code from ASCET and Simulink models. Modules may be combined to form "Functions" - which represent specific features of the overall control system, such as idle, EGR, ABS or spark control. Functions and Modules combine to form "Software Systems" - which represent the entire set of features for a specific domain (e.g. engine or chassis). A SiL system is then created by assigning the Software System to run on the PC hardware. This assignment automatically invokes the appropriate compiler tool chain and libraries for PC execution.
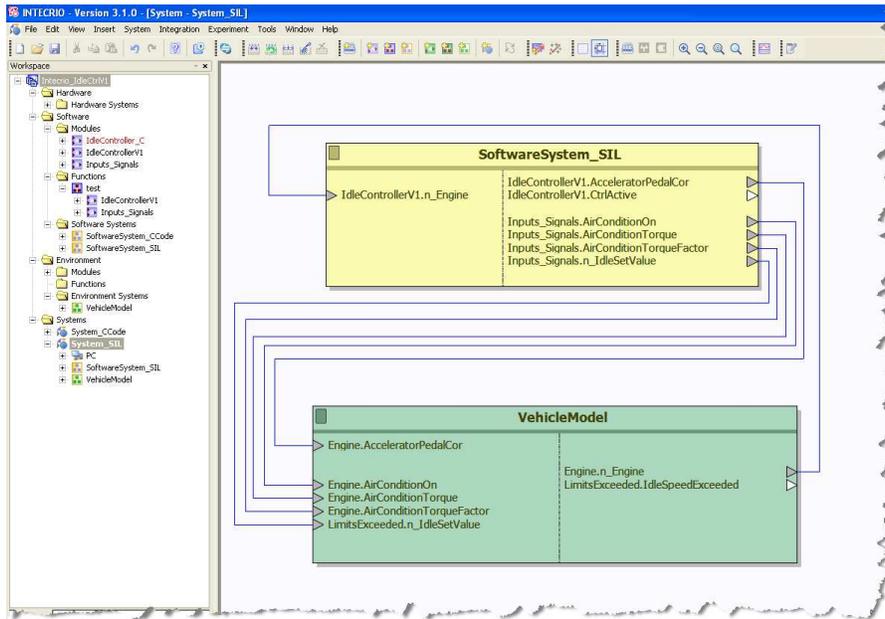
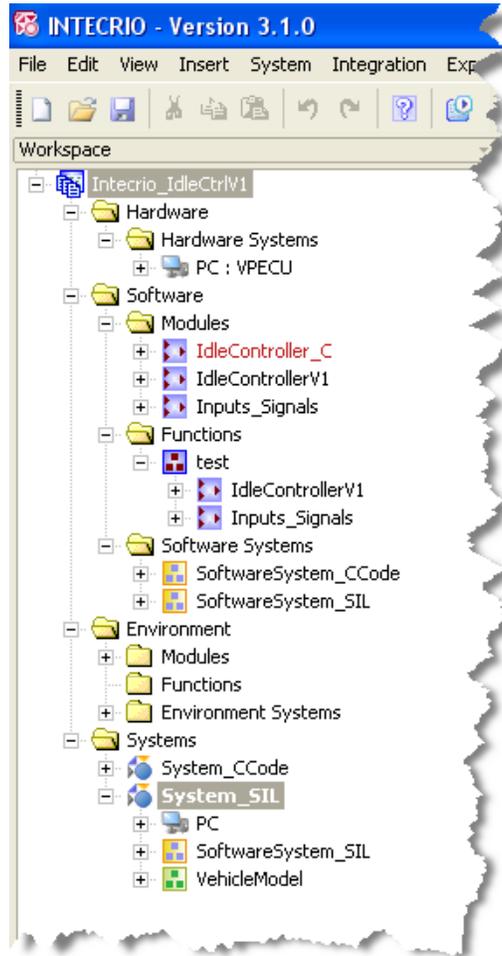Figure 10: A SiL system set up in INTECRIO

Figure 11: Software Architecture Emulation in INTECRIO

Since a SiL system runs on the PC, there are no real-time or memory constraints. INTECRIO offers optimum utilization of the available horsepower of the PC by providing a mechanism to control the timescale of the simulation during run time (Figure 12). With this mechanism, users can either scale the simulation (speed up or slow down) according to their needs, or simple adapt to run it as fast as the PC will allow.
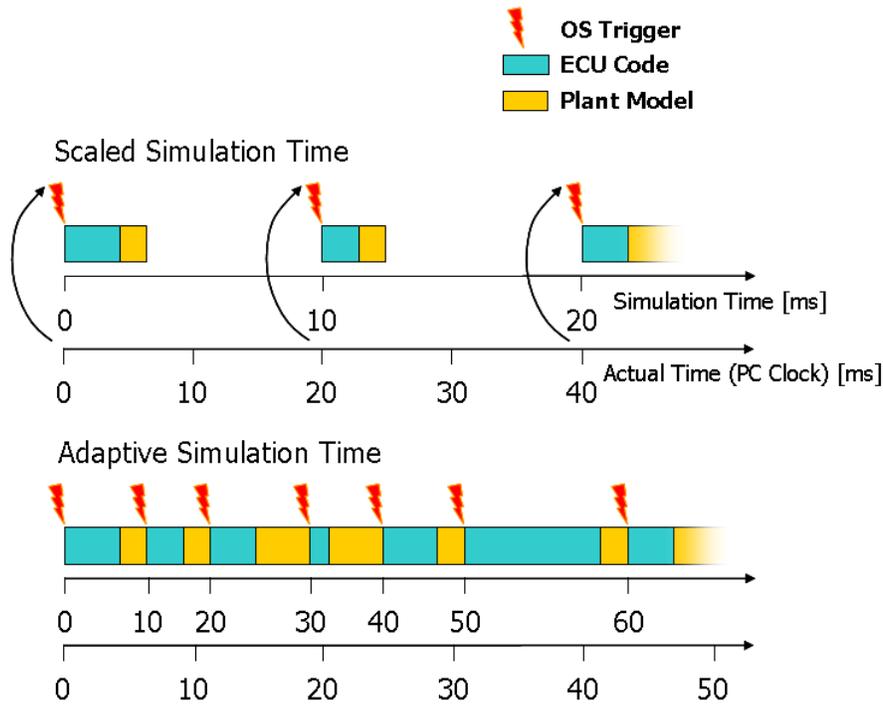

Figure 12: Scaled and Adaptive Simulation Time in INTECRIO

If the plant model cannot be integrated within INTECRIO, a co-simulation may be set up (as in Figure 5). This is useful for variable step-size simulations and for domain specific models in that execute in proprietary environments. The ECU C-code and model generated code is still assembled via INCODIO into the INTECRIO environment.

Once the SiL system is compiled by INTECRIO, a software image (executable) for Windows and a corresponding ASAP2 file are created. Using the INTECRIO Experiment Environment (Figure 13), engineers can easily execute the SiL system on the PC, create graphical user interfaces (monitoring and control widgets, oscilloscopes, displays etc.) and log data from their tests. The experiment environment contains a large library of pre-defined widgets for a multitude of applications, and users may create their own .NET compatible widgets. One of the major advantages offered by this user interface is an open COM API that allows engineers to create platform independent scripts that allow the automatic execution of tests and collection of data [5] for verification and validation purposes.
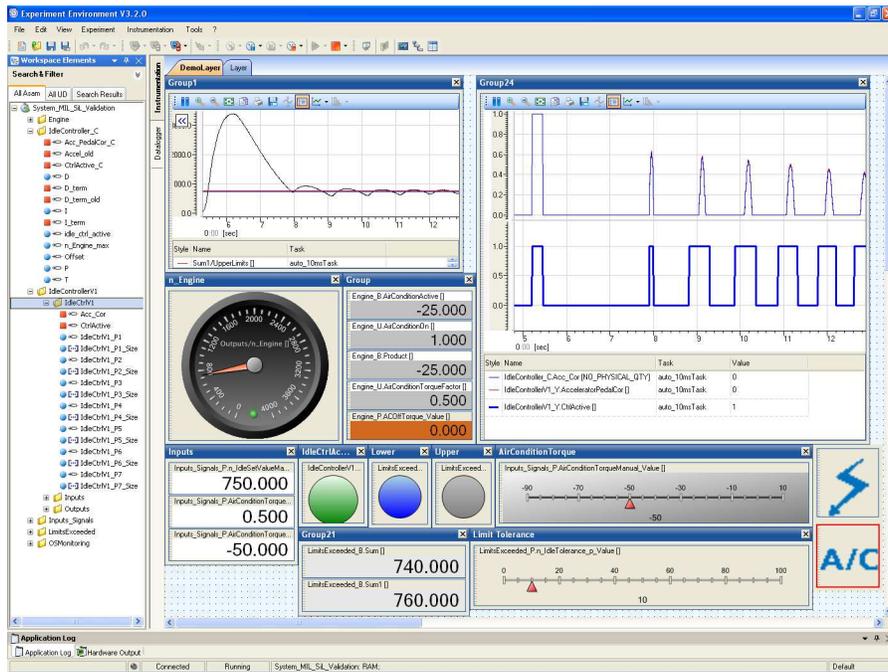
Figure 13: INTECRIO Experiment Environment

After a SiL system is validated on the PC, INTECRIO can compile the same code to execute in real-time on an ETAS rapid prototyping target such as the ES1000 and ES910 [9]. The advantage of this extension is that the C-code can now interface with a range of real-time signals (e.g. analog, digital, PWM, CAN bus) in a dynamometer or a vehicle (Figure 14). Since the ETAS rapid prototyping targets use the same RTA-OSEK operating system as the PC target, there is no change in the configuration of the software system.
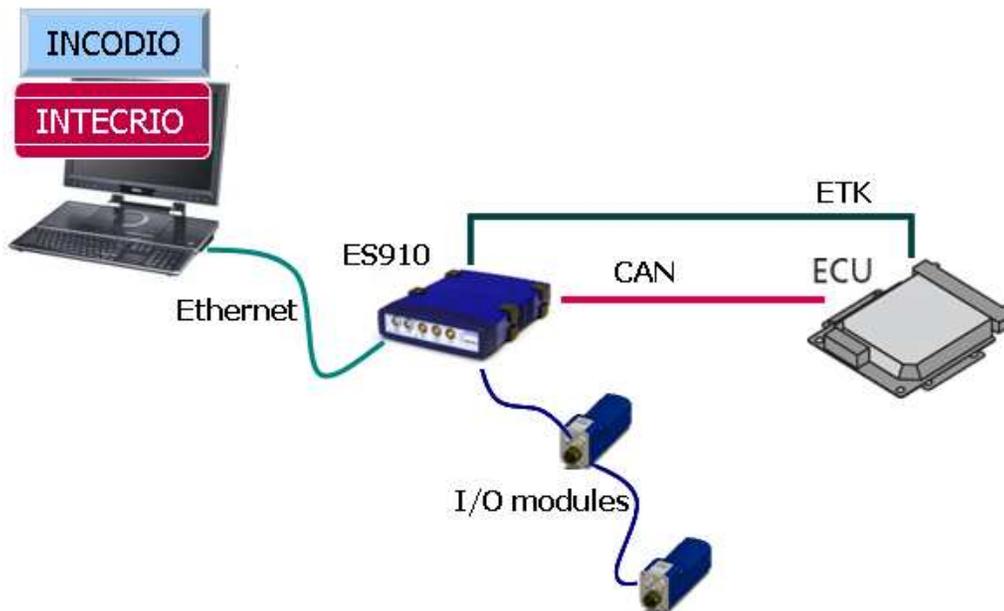

Figure 14: Rapid Prototyping with C-code using INTECRIO and INCODIO

SIL APPLICATIONS

C-code validation on the PC

In a number of projects, minimal changes to existing software are needed for creating a new variant of the ECU (e.g. for a smaller vehicle, reduced feature set or for relaxed regulatory requirements). In such cases, it is customary to make the necessary changes or write additional code using a commercial off-the-shelf C-code editor (commonly known in the industry as an Integrated Development Environment, or IDE, for C-code). There is no new model generated code to integrate. In this case, a SiL system (composed only of C-code modules) is very useful to quickly validate the functionality of the code changes on the PC before ECU hardware is available.

Using INCODIO, first a single software component is integrated and simulated with INTECRIO. Then, additional components are added until a complete ECU system is created in a step-by-step manner. During this process, the simulated software components will most likely contain references to other C-code files, functions and variables, which are not yet included in the system. INCODIO uses a special reference resolving technology to search for the missing information in the available ECU software repository (typically thousands of files for a typical ECU project) and create the necessary definitions to close such references. Once the complete system is defined in INCODIO and integrated using a virtual OS in INTECRIO, the code can be simulated, its features evaluated and calibrations tweaked to meet the necessary performance requirements. If a plant model is not available, stimuli or test data may be used to run the functional tests on the code. Since the SiL system in running on the PC, access to the PC-based file system allows file I/O for reading/writing data and debug messages.

Virtual Validation of C-code with Models

As mentioned earlier, a majority of new projects are composed of a few new ECU functions that are created in model-based environments - such as ASCET, Simulink/Stateflow and others. In the early phases of development, these models can be independently verified using desktop simulation in their native environment. However, legacy C-code and generated code from other tools is often needed to create a fully functional ECU system on the PC. Together with appropriate plant models, a "system simulation" of the complete vehicle can then be realized. The benefits of establishing such a system have already been presented earlier.

Using INTECRIO, the model-based components from ASCET and Simulink may be combined with C-code components from INCODIO to form the ECU part of the system. After scheduling the software components on the virtual OS, INTECRIO compiles the complete virtual system and creates a PC compatible executable along with a signal description file (ASAP2 format) that contains all the measurement and calibration elements of the code and models. The INTECRIO Experiment Environment can be used to control the execution of this system, conduct virtual tests (e.g. automated regression tests, plausibility tests etc.), change calibrations and collect data.

In some applications, the functional behavior of the ECU software running on a PC must be compared to the behavior observed in an original production ECU. This can be accomplished by first recording the relevant input and output signal data of a software component (or the entire system) during a field test and then automatically creating the stimuli signals for the SiL system using INCODIO. INTECRIO executes the SiL system using these stimuli signals and records the outputs on the PC. Once the outputs of the SiL system match the outputs observed in the vehicle, the ECU software on the PC is considered functionally equivalent. A plant model (or a set of plant models) may now be added to the SiL system. Fidelity of the plant model is easily checked using the previous measurement data as well. This allows the SiL system operator to fine tune the

plant model until there is a close match with the measurements in the vehicle. Once this is achieved, the SiL system closely resembles the real vehicle, and is ready for further feature development or closed-loop testing.

One major advantage of this system is that, once created, the INTECRIO PC executable can be deployed on multiple PCs in order to allow test engineers to test or validate different parts of the software that they are responsible for. These users do not need access to the source code files, models or even the software licenses for the modeling tools themselves. The INTECRIO Experiment Environment is all they need to "run" the executable to simulate the system on their PC. This workflow enables a new business model for desktop software development - where only a few system experts are necessary to create various system simulation packages, while a multitude of test engineers can conduct verification and validation tests on their PCs without having to deal with the complexity of the SiL setup.

Virtual calibration on the PC

One of the major tasks prior to release of ECU embedded software is the proper calibration of all functions to meet performance and drivability requirements. While certain calibrations will always be evaluated best in a vehicle (e.g. transmission shift quality, drivability etc.), a large number of calibrations can be developed in the virtual environment, saving a lot of time and effort later. Due to reductions in available test vehicles, most manufacturers are now in favor of conducting a bulk of the calibration work in the lab or in the desktop environment. The SiL system offers an ideal setup for this purpose. Once the C-code is assembled by INCODIO and integrated for simulation by INTECRIO, standard calibration tools like ETAS INCA [9] can be used on the desktop for calibration as well as measurement and recording of code variables. For this purpose, INTECRIO generates a full ASAP2 file containing all the C-code parameters identified in INCODIO for measurement and calibration on the PC. This eliminates the need for proprietary solutions that are normally needed in custom SiL environments for this purpose. With INCA, a complete ECU calibration set can be stored in a data file (ETAS DCM format) and uploaded to the SiL system to initialize the software correctly. Changes in calibration during a simulation can be stored in a new calibration data file and uploaded next time the simulation is started. Automatic calibration methods can also be used via the INCA COM-API.

Migration to AUTOSAR

To enable a smooth transition to the AUTOSAR software architecture, INCODIO supports the AUTOSAR interface description format (stored as an XML file). If hand-written C-code is modified to comply with AUTOSAR, such an interface description file is necessary for each software component. INCODIO then parses the AUTOSAR XML file to create the complete AUTOSAR software component description for simulation by INTECRIO (including the information for the input and output ports and runnables). INTECRIO can import the software component directly from INCODIO. In addition, INTECRIO can import AUTOSAR software components created by MBD tools (e.g. ASCET) with their associated description files.

Since migration to AUTOSAR is a step-by-step approach, INTECRIO is designed keeping this need in mind. Therefore, AUTOSAR software components can be mixed with standard software components in INTECRIO. The built-in AUTOSAR RTE generator then generates the necessary glue code to manage the transfer of data across the interfaces. Thus, functional validation of an AUTOSAR ECU system is possible on the PC (Figure 15). This is a huge advantage because it eliminates the need to wait for an AUTOSAR compliant ECU for doing the software validation. Moreover, most migration issues in the application software can be resolved on the PC.
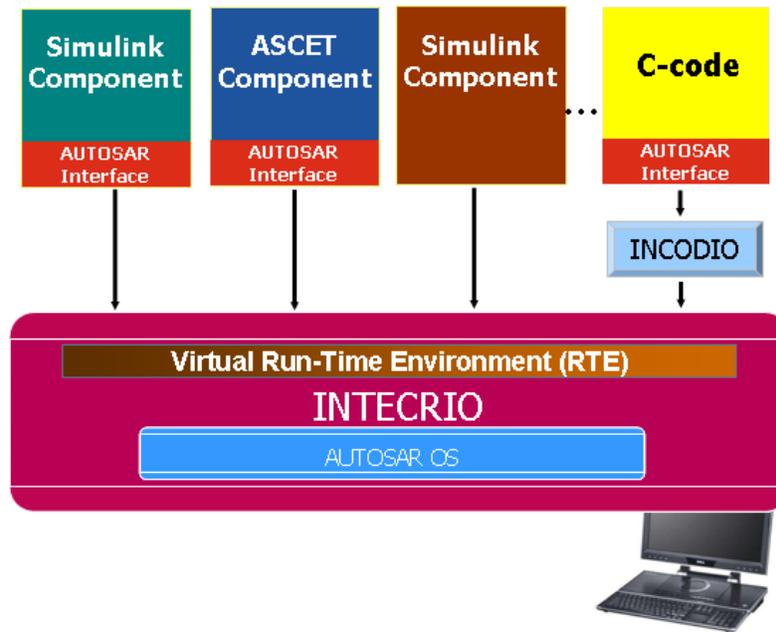
Figure 15: AUTOSAR validation using INTECRIO

CASE STUDY

Several OEMs and ECU suppliers are currently utilizing the tool chain above to migrate their ECU development process to the PC. In this section, we describe how PSA Peugeot Citroën (PSA) is accelerating their AUTOSAR software component validation process using INTECRIO, INCODIO and INCA.

CHALLENGE:

In a strategic ECU development project, PSA is developing advanced powertrain ECU functions in the Simulink environment. These models are converted to c-code by their ECU suppliers, who treat the Simulink models as a graphical specification. The ECU suppliers also create AUTOSAR 2.0 compliant interfaces for all software components. Once these components are shipped back to PSA, the challenge is to validate the functionality of these components against the original Simulink specification. Further, PSA wanted to calibrate the AUTOSAR software components on the PC using the established calibration tool INCA.

REQUIREMENTS:

Since there are several ECU suppliers creating software for PSA, the first requirement was a modular and scalable SiL platform that did not require any change in the structure of the software components delivered by the ECU suppliers for validation purposes. Secondly, it was necessary to have a well defined calibration process for AUTOSAR 2.0 compatible components. PSA also required rapid SiL construction and validation after receiving new software components from their ECU suppliers. Finally, the solution needed to be compatible with ongoing changes in the AUTOSAR specification.

SOLUTION:

PSA has deployed the ETAS/SYSTECS solution for SiL validation described in this paper. Figure 16 shows the schematic of the workflow. With INCODIO, the AUTOSAR software components delivered by the ECU

suppliers are easily synthesized into INTECRIO modules with the help of the delivered interface description files (xml format). The original Simulink models and the AUTOSAR software components are compiled into a single PC executable by INTECRIO. INCA is then used to validate and calibrate the software on the PC. With the help of this tool chain, PSA engineers are able to integrate c-code and conduct validation tests typically in a couple of hours as compared to a couple of days that it took to integrate the code into Simulink S-functions. Moreover, the SiL system now runs in near real-time, saving a tremendous amount of time in the validation and calibration process.   Since there is no change required to the code from the ECU suppliers, the quality of the technical exchange with the suppliers has gone up and the first-pass software quality has shown big improvements.
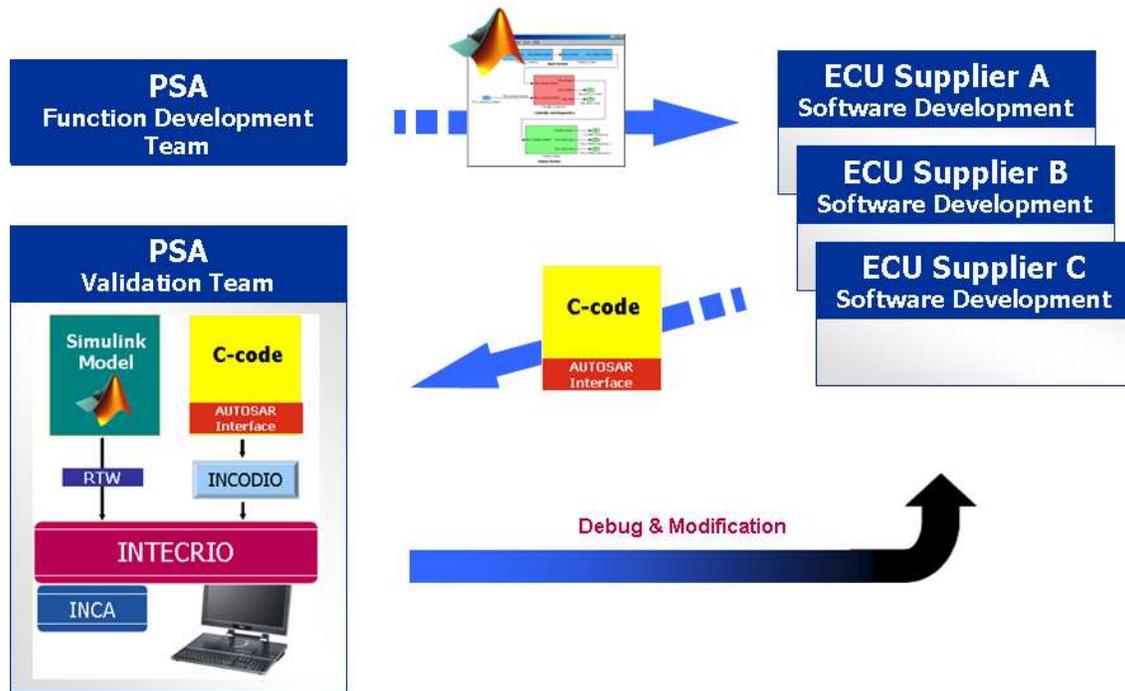

Figure 16: PSA SiL Validation for AUTOSAR

CONCLUSION

Although PC-based development is gaining higher levels of importance in the development of new ECU software, traditional model-based tools are found not to be sufficiently prepared to meet the challenge. This is particularly true when legacy C-code has to be included in the PC-based environment. Many proprietary solutions [4][6] have been created by companies to address this challenge. While great at solving particular problems, proprietary solutions are difficult to maintain and support.

ETAS and SYSTECS have partnered to create an innovative and flexible framework for creating PC-based simulations of C-code for testing and calibration purposes. This solution has already been deployed successfully in large scale powertrain ECU programs by major auto manufacturers. The open framework lends itself to easy customization to each customer's specific needs and continues to be enhanced to meet the broader industry requirements, as demonstrated by the AUTOSAR use case.

REFERENCES

1. V. Jaikamal, ETAS Inc, "Model-Based ECU Development – An Integrated MiL-SiL-HiL Approach", 2009 SAE World Congress, Detroit, MI, 2009-01-0153
2. T. Zurawka, O. Meyer, SYSTECS Informationssysteme GmbH, "INCODIO – A new tool for automotive Software-in-the-Loop applications", 2007 Internationales Stuttgarter Symposium (Automobil- und Motorentechnik), Stuttgart, Germany
3. W. Eismann, J. Wagner, ETAS GmbH, "INTECRIO v3.0 – Open for Business. And for AUTOSAR.", ETAS "Real-Times" magazine, Issue 1-2008, pp 10-12.
4. K. Lang, K.J. Mitts, General Motors Corporation; T. Roudier, D.L. Kiskis, ChiasTek Inc., "Using a Co-Simulation Framework to Enable Software-in-the-Loop Powertrain Systems Development", 2009 SAE World Congress, Detroit, MI, 2009-01-0520
5. P. Vuli, M. Badalament, V. Jaikamal, ETAS Inc., "Maximizing Test Asset Reuse Across MiL, SiL and HiL Development Platforms", 2010 SAE World Congress, Detroit, MI, 2010-01-0660
6. V.R. Nallapa, F. Syed, F. Jiang, S. Semenov, Ford Motor Company, "Automated Migration of Legacy Functions and Algorithms to Model Based Design", 2008 SAE World Congress, Detroit, MI, 2008-01-0747
7. O. Philipp, M. Buhl, S. Diehl, M. Huber, S. Roehlich, J. Thalhauser, TESIS DYNAware GmbH, "Engine ECU Function Development Using Software-in-the-Loop Methodology", 2005 SAE World Congress, Detroit, MI, 2005-01-0049
8. M. Sunwoo; J. Han, J. Ma, J. Youn, Hanyang University, "Validation of a Seamless Development Process for Real-time ECUs using OSEK-OS based SILS/RCP", 2008 SAE World Congress, Detroit, MI, 2008-01-0803
9. ETAS Tools (INTECRIO, ASCET, INCA, RTA-OSEK, ES910, ES1000), www.etas.com
10. Simulink, The Mathworks, www.mathworks.com
11. ASAM-MCD-2MC, www.asam.net
12. AUTOSAR, www.autosar.org
13. OSEK-VDX, www.osek-vdx.org

CONTACT INFORMATION

Vivek Jaikamal is a Marketing Manager for software engineering tools at ETAS Inc., Ann Arbor, MI, USA. He may be reached at vivek.jaikamal@etas.com.

Thomas Zurawka is Managing Director at SYSTECS Informationssysteme GmbH, Stuttgart, Germany. He may be reached at thomas.zurawka@systecs.com.

ACKNOWLEDGMENTS