A Modular Framework for Modeling Hardware Elements in Distributed Engine Control Systems

Alicia M. Zinnecker *
N&R Engineering, Parma Hts, OH, 44130, USA

Dennis E. Culley †
and
Eliot D. Aretskin-Hariton ‡
NASA Glenn Research Center, Cleveland, OH 44135, USA

Progress toward the implementation of distributed engine control in an aerospace application may be accelerated through the development of a hardware-in-the-loop (HIL) system for testing new control architectures and hardware outside of a physical test cell environment. One component required in an HIL simulation system is a high-fidelity model of the control platform: sensors, actuators, and the control law. The control system developed for the Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k) provides a verifiable baseline for development of a model for simulating a distributed control architecture. This distributed controller model will contain enhanced hardware models, capturing the dynamics of the transducer and the effects of data processing, and a model of the controller network. A multilevel framework is presented that establishes three sets of interfaces in the control platform: communication with the engine (through sensors and actuators), communication between hardware and controller (over a network), and the physical connections within individual pieces of hardware. This introduces modularity at each level of the model, encouraging collaboration in the development and testing of various control schemes or hardware designs. At the hardware level, this modularity is leveraged through the creation of a Simulink® library containing blocks for constructing smart transducer models complying with the IEEE 1451 specification. These hardware models were incorporated in a distributed version of the baseline C-MAPSS40k controller and simulations were run to compare the performance of the two models. The overall tracking ability differed only due to quantization effects in the feedback measurements in the distributed controller. Additionally, it was also found that the added complexity of the smart transducer models did not prevent real-time operation of the distributed controller model, a requirement of an HIL system.

Nomenclature

Variables

N	shaft speed
P	pressure
P_{50}	pressure at station 50 (exit of low-pressure turbine)
T	temperature
T_s	update rate/sampling time
W	flow rate
$(\cdot)_{sens}$	sensed variable

*Control Systems Engineer, alicia.m.zinnecker@nasa.gov, AIAA Member.
†Research Engineer, dennis.e.culley@nasa.gov, AIAA Senior Member.
‡Research AST, Control Systems, eliot.d.aretskin-hariton@nasa.gov

Acronyms

ADC	Analog-to-Digital Converter
C-MAPSS40k	Commercial Modular Aero-Propulsion System Simulation 40k
CM	Controller Model
DAC	Digital-to-Analog Converter
DEC	Distributed Engine Control
DECWG™	Distributed Engine Control Working Group
ECU	Engine Control Unit
EM	Engine Model
FMV	Fuel metering valve
HIL	Hardware-in-the-Loop
NCAP	Network Capable Application Processor
psi	pounds-force per square inch
s	seconds
SC	Signal Conditioning
S/D P	Signal/Data processing
STIM	Smart Transducer Interface Module
TEDS	Transducer Electronic Data Sheet
WM	Wrapper Model
XDR	transducer
V	volts
VBV	Variable Bleed Valve
VSV	Variable Stator Vanes
μP	application processor (microprocessor)

I. Introduction

ALTHOUGH widely-used in the automotive industry, distributed control has been slow to transition into the aerospace industry because the challenges related to implementation of such architectures are perceived to outweigh the benefits.[1,2] In general, one characteristic differentiates distributed control from centralized control: the ability to spread computational capability across the control system. In a centralized architecture, the control algorithm resides in the engine control unit (ECU), along with data conversion functionality. Although necessary for the controller to be able to use sensor measurements (or for the actuator to respond to a control command), there is no requirement that conversion between analog and digital domains be handled by the ECU. In fact, if the capability existed, these tasks could be off-loaded to processors local to the sensor and actuator hardware, freeing the ECU to commit its processing power to the evaluation of the control algorithm.

Additional benefits of distributed control are realized through the use of a network for communication of digital data between the ECU and effectors. The digital network replaces the multitude of individual wires present in a centralized architecture with a bus that is less complex and lighter weight. Additionally, the network provides a common physical interface between the hardware and control elements, promoting modularity of components and reusability of code. Having the ability to replace or upgrade one element of a control system, such as a sensor, may encourage the development of a 'partial' certification process, wherein the entire control system, once certified, does not need to be recertified after a hardware change. Instead, it would only be necessary to certify the new hardware, since it is interfacing with the rest of the system in the same way as the old hardware. As this technology is reused in several applications, performance data can be compiled to provide an indication of the expected performance of the control system. Modularity also presents benefits related to fault isolation and the logistics of maintenance over the life cycle of an engine.[1,2]

Several obstacles must be overcome before the benefits of distributed control can be realized in an aerospace application. One of the most substantial barriers slowing the progress toward adoption of distributed control in an engine is related to the desire of engine and hardware manufacturers to protect intellectual property. Although engine companies and parts manufactures currently collaborate in producing engine parts to specification, these partnerships are limited due to the desire for maintaining a competitive edge in the market. The Distributed Engine Control Working Group (DECWG™) provides a forum for companies and developers to contribute ideas in pre-competitive areas to advance development of technology

related to distributed control without compromising proprietary information.[1,2] Two areas of interest to DECWG include high-temperature electronics and a network to allow communication between the hardware and the ECU. These represent additional hurdles to the implementation of distributed control as, without electronics that can withstand the harsh environment of the engine and a reliable means to relay feedback measurements to the controller or commands to the actuators, distributed control is not a viable architecture in an aerospace engine. While the concept of data being communicated over a network in a control system is by no means novel, the specific needs of such a system in an aviation application require special attention to be paid to this element.[2] Two network-related mechanisms, time delay and packet loss, may have significant effect on the reliability and stability of the closed-loop system if not properly taken into consideration during control design. These mechanics have been studied and the results of these investigations offer initial impressions as to the importance of modeling the network with the controller.[3–7]

A final, more encompassing obstacle that must be addressed is related to the design process as a whole. Control design, testing, and implementation are often the final steps in the engine development process and are therefore subject to strict time and budget constraints.[8] The high risk of developing new technologies and implementing new ideas tends to impede progress toward alternative control architectures over such a limited time frame. Having the ability to conceptualize and develop the system architecture independent of a physical engine prototype would allow for relaxation of many of these project constraints, particularly that of time, leading to implementation of distributed engine control in an aerospace application.[8]

A hardware-in-the-loop (HIL) system is under development at NASA Glenn Research Center, as part of the Distributed Engine Control (DEC) task, which aims to provide a simulation environment for testing control architectures and hardware without the need for a physical engine. This effectively allows the control design to develop in parallel with engine design. Performing computational studies with this system will provide an understanding of the capabilities of the control system while reducing the cost and risk associated with testing new hardware and control configurations in a physical test cell. In addition, simulations may be run to model engine behavior at flight conditions too extreme to achieve in a test cell, allowing for the physical limitations of the hardware and controller to be studied. This kind of simulation requires high-fidelity computational models of the engine and of all elements comprising the control platform (sensors, actuators, controller, and controller network). The hardware models should capture local data processing effects in addition to sensor and actuator dynamics, and the network model should impose realistic data delay or loss as packets are transferred between the hardware and controller. With these enhanced models, results from simulation of the controller model with computational hardware models can be expected to be similar to those from simulation of the controller model with hardware prototypes in the control loop.

To facilitate the development and testing of controller hardware models and control algorithms in the HIL system, a modeling framework has been developed that relies on the definition of consistent interfaces between elements of the control system. These interfaces define clear delineations between components at several levels of the model: between the engine, controller, and user input; between the elements of the controller; and between the components within each control element. Although they may be defined on a model-to-model basis, these interface definitions enable modular development and expansion of the controller model within the overall system. A Simulink library is under development to provide elements for constructing hardware and network models with this flexibility and added fidelity.

This paper is organized as follows. In Section II, the structure of the control system in the Commercial Modular Aero-Propulsion System Simulation, 40k (C-MAPSS40k), which is the baseline for the framework developed in the ensuing discussion, is presented. Emphasis is placed on the modifications necessary to realize a distributed control architecture. Section III adds details to the modular framework related to implementation of these changes, including a brief discussion of the modeling and simulation trade-offs associated with such an approach. The functionality and implementation of the Smart Transducer Library is introduced in Section IV, with comparison of the results from simulations of controllers with different architectures. Consideration is given to the practicality of this added fidelity in a controller model running in real-time as part of an HIL simulation. The paper concludes with a summary of the modeling framework and a brief outline of future development of the HIL system around this framework.

II. Structure of the baseline controller model

In general, there is no required framework to which the closed-loop engine model in an HIL system must conform; for the proposed modeling approach, the configuration in Fig. 1 is chosen. The three main

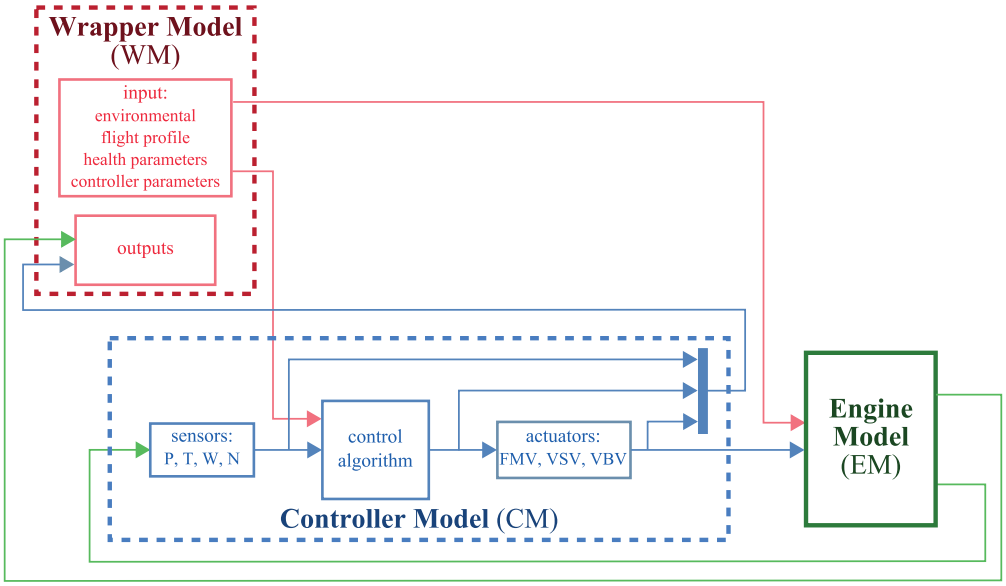

Figure 1. Block diagram showing the general framework of the HIL simulation system. The sensed values (pressure (P), temperature (T), flow rate (W), and shaft speed (N)) and actuators (fuel metering valve (FMV), variable stator vane (VSV), and variable bleed valve (VBV)) listed in the figure are specific to the C-MAPSS40k controller.

components within this framework, as indicated in the figure, are an engine model (EM), a controller model (CM), and a wrapper model (WM). Each model is self-contained with well-defined inputs and outputs for establishing connections and running a simulation. The WM governs a simulation by providing environmental and health inputs to the engine and controller models, and collecting operating data from these models for analysis. The dynamic response of the engine is calculated by the EM and used by the CM, which models hardware dynamics and the controller algorithm. (For HIL simulations, hardware connected in the control loop replaces the corresponding model in the CM.) The HIL system under development at NASA represents one physical implementation of this framework; the details of this system are outside the scope of this paper and may be found in Refs. 9, 10.

The CM is further structured by defining three subsystems (Fig. 1): one containing actuator models, one containing sensor models, and one implementing the control algorithm. The baseline C-MAPSS40k controller model, referred to as the 'unstructured' model throughout the paper, models only the dynamics of the sensors and actuators along with the control law, and needed to be restructured to fit this framework. This required grouping the relatively simple first-order transfer functions modeling the pressure (P), temperature (T), flow rate (W), and shaft speed (N) sensors in one subsystem, those modeling the fuel metering valve (FMV), variable stator vanes (VSV), and variable bleed valve (VBV) actuators in another, and the blocks performing control calculations in a third. Three levels of interfaces are created in the CM by this structure: the top level communication with the WM and EM, a middle level for communication between sensors, actuators, and the controller, and the lower level representing data transmission within individual hardware models and the control algorithm.

This hierarchy of interfaces outlines a modular modeling approach through which a distributed control architecture can be introduced to the CM. Modularity requires consistency in the interfaces at each level of the model to ensure the number of input and output ports does not change when a hardware model, or other subsystem, is modified. (Other attributes, such as the number of elements in the data arrays passing through these ports, are model-specific and should be handled on a case-by-case basis.) This reduces the number of changes necessary to the overall model when a hardware model is updated. Having a standardized interface also enables the independent development of hardware models and the control algorithm, providing greater opportunity for collaboration.

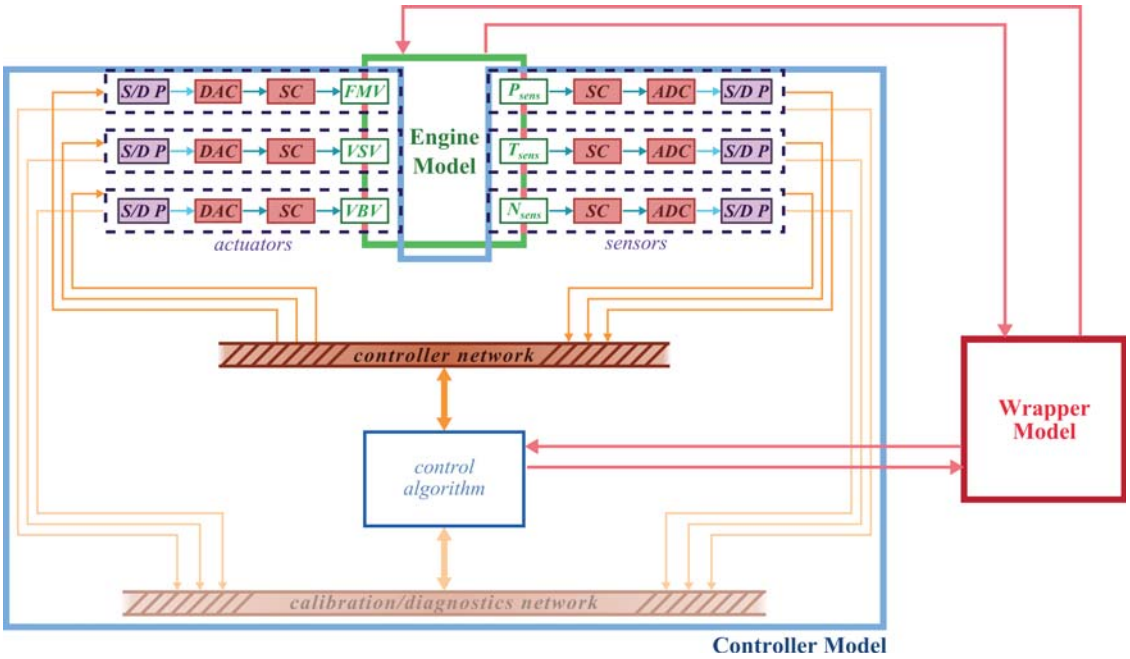

Figure 2. Diagram of the interfaces in a distributed control system: data flow is indicated by arrows and physical interfaces (such as between the engine and sensors, $(\cdot)_{sens}$ by overlapping blocks. Blocks with colored backgrounds are unique to this architecture: the controller network and the signal conditioning (SC), conversion (ADC, DAC), and signal/data processing (S/D P) functionality on the hardware. The calibration network, and its connections, are in lighter colors to indicate future functionality.

III. Approach to modeling distributed control system

Simulation of a controller with a distributed control architecture may be constructed by incorporating a model of the digital controller network in the CM, and improved accuracy may be achieved by developing higher fidelity hardware models to use in the simulation. The presence of these new components is expected to improve the realism of the overall system and, if developed appropriately, will enable the modeling of failures. Figure 2 indicates how these components fit within the HIL framework. Pink arrows in the figure represent the top-level interfaces (between the EM, CM, and WM) while orange arrows show the interfaces within the CM (between controller and hardware elements). The blue arrows are the lowest-level interfaces in the model, between the components that comprise each individual piece of hardware. A second set of network interfaces, not part of the framework in Section II, establishes a redundant link between the controller and control elements that may be used to diagnose hardware failures. These are shown in lighter colors in Fig. 2 as the modeling of failures in the system, and the use of diagnostic information to identify these failures, is outside the scope of this paper.

A. Modeling the controller network

Within the proposed modeling framework, a controller network replaces the direct interface between subsystems at the middle level of the model hierarchy. Adding the controller network to the model introduces an important feature of a distributed control system: the means by which data are transferred between the hardware and controller. The network acts to ensure data written by a specific sensor (or by the controller) are routed correctly and are available to be read by the controller (or an actuator). Ideally a direct feedthrough (as in the C-MAPSS40k controller), this data transmission can have a negative effect on the closed-loop performance when an actual network is used. Adding the fidelity of data delay or loss allows the computational model to better reflect the expected performance, reliability, and stability of the same controller with physical hardware in the loop and a real controller network for communication. In addition, the presence of the controller network model reinforces the 'decoupling' between the hardware and the controller in the physical system where only the presence of data on the network matters to the controller or actuator, not the specific hardware that placed the data there. This separation maintains

modularity within the model and enables easy expansion of the system: hardware may be replaced or added as nodes of the network with minimal change to the rest of the system.

B. Modeling the sensors and actuators

At the lowest level of the modeling framework are individual models of each sensor and actuator. In a system with a distributed control architecture, data processing functionality resides local to the sensor or actuator hardware, forming a 'smart' transducer. (Throughout this discussion, the term 'transducer' is used to refer to either a sensor or an actuator.) Because the CM is intended to support both computation and HIL modeling of a control system, the sensors and actuators should be modeled as smart transducers. Although no universal standard exists for smart transducers, the IEEE 1451 specifications have been developed to provide a set of guidelines for smart transducers with 'plug-and-play' components.[11–17]

A smart transducer that is compliant with the IEEE 1451 guidelines has the structure shown in Fig. 3. The Smart Transducer Interface Module (STIM) is composed of the transducer hardware and components performing signal conditioning and conversion between the analog and digital domains. All of the signals in the STIM are analog, except for the signals that interface with the Network Capable Application Processor (NCAP), where a microprocessor and a network adapter are located. Programmed within the STIM is a Transducer Electronic Data Sheet (TEDS), which contains identification and calibration data for the transducer hardware, and is accessible by the NCAP during data processing. The components in a smart transducer are independent of each other and can therefore be combined or replaced easily because the interface definitions are formalized in the IEEE 1451 specification. This modularity has inspired a similar approach for modeling such hardware.

The Simulink environment, in which components of the HIL system are implemented, allows for the development of user-defined libraries. This feature can be leveraged as part of the modeling framework by creating a model library containing blocks for modeling in an IEEE 1451-compliant smart transducer. In the ensuing discussion, the term 'library' is used to refer to the total collection of blocks, and 'sublibrary' refers to a specific subset of these blocks, as illustrated in Fig. 4. In the Smart Transducer Library, blocks are organized into sublibraries reflecting the functionality of the components in Fig. 3: hardware models, signal conditioning and conversion, data processing, and network models. Details of the use of this library will be provided in the next section.

The modeling framework introduced in Section II is shown with added detail in Fig. 5. The Smart Transducer Library offers models of components that can be combined to form smart transducer models that in turn may be incorporated in the CM. As the library is further developed, assuming common interfaces for blocks in each sublibrary, the fidelity and functionality of the smart transducer models can be improved by simply replacing the old block and reconnecting the input and output ports. Additionally, this framework enables collaboration at several levels, wherein one collaborator may focus on constructing the Simulink library, another collaborator may use these blocks to model real hardware or the controller network, and a third may concentrate on implementation of the control law in Simulink. The common interfaces allow these components to be integrated into a single CM with little problem, as each collaborator knows the expected inputs and outputs of the block they are to contribute.

C. Trade-offs associated with this hierarchical framework

As with any design choice, a set of trade-offs must be evaluated for this modeling framework. Many of these trade-offs are related to the establishment of consistent interfaces at all levels of the hierarchy illustrated in Fig. 5. At the highest level, these interfaces act to decouple the CM, EM, and WM by requiring that elements in the data arrays passed between these models are specified so that they can be

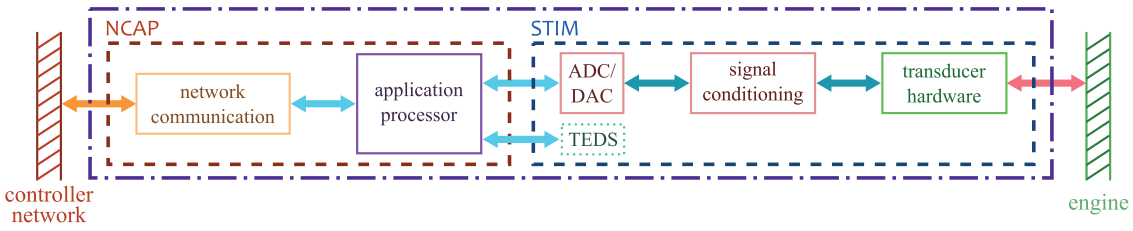

Figure 3. Block diagram of a smart transducer complying with the IEEE 1451 specifications.

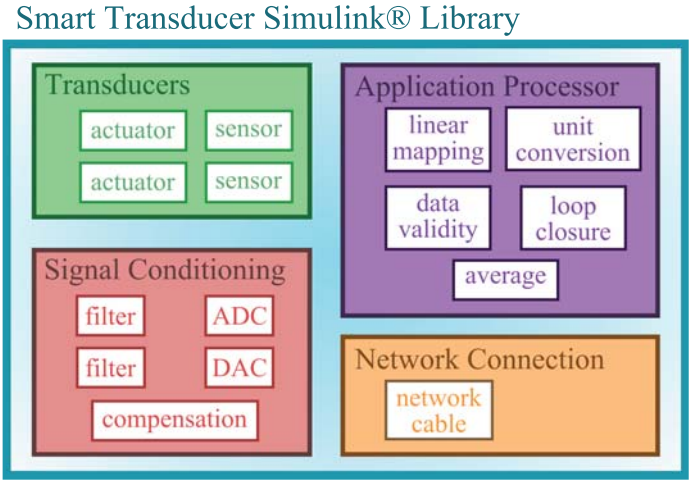

Figure 4. Organization of the blocks and sublibraries that comprise the Smart Transducer Simulink library.

routed correctly. Within the CM, the presence of the controller network effectively 'separates' the sensor, controller, and actuator models, requiring only definition of the physical interface between these components and the network to be defined. Extended to the lowest level of the hierarchy, ensuring the correct data is available at the input port of each component of an individual smart transducer allows blocks from the Smart Transducer Library to be connected in any order to create a smart transducer model. The modularity resulting from such an architecture offers a mix of trade-offs between ease of collaborative development, speed of simulation, and complexity of the model.

The most immediate benefit afforded by this approach is that, by having consistent interfaces between each element of the CM, models of varying fidelity may be developed independently. These models can then be integrated in the CM by routing the inputs and outputs appropriately. This enables the easy exchange of different controller, engine, and smart transducer models within the system. Additionally, elements of the CM may be replaced by compiled s-functions, provided the inputs and outputs are consistent with the interface definitions, so that proprietary hardware, control laws, or engine models may be incorporated in the simulation without knowing the details of the implementation. The flexibility related to this modularity becomes limited at higher levels of the hierarchy, however, due to the larger amount of data transferred between, for example, the CM and EM than within a smart pressure sensor. Some flexibility can be gained by defining the interface in general terms (e.g., identifying a block of elements as containing pressure data instead of relating a single element with pressure at a specific location). Even so, the developers of the CM and EM must still communicate to ensure the data arrays produced by each model are properly ordered and that the input data arrays are correctly unpacked. No matter how modular a system is, this limitation will always exist.

A second set of trade-offs relates to the ability to utilize model libraries in constructing sensor and actuator models of varying fidelity. In this particular application, a library is under development containing blocks for modeling smart transducers based on the IEEE 1451 specification. The use of a library facilitates upgrades to the hardware models and simplifies development of these models. Instead of starting from scratch in constructing a smart transducer model, the physical dynamics of the hardware and the effects of filtering and other processing are each modeled by blocks in the library that may be combined to form the complete model. Components may then be updated individually, as the library develops. Having more accurate models, however, increases the complexity of the system in two ways: by providing information that is not needed by the control algorithm and requiring longer computational time. The additional information, such as timing information in a network model, is not needed to calculate control demands, but is required to improve the realism of a distributed controller model. The fidelity of the hardware models is coupled with a common trade-off in design problems: as the sensor or actuator model becomes more complex and, as a result, more accurate, it may prevent real-time simulation of the CM, a requirement if hardware prototypes are to be simulated in the control loop. This may be mitigated by improving the computational platform on which a CM is simulated.

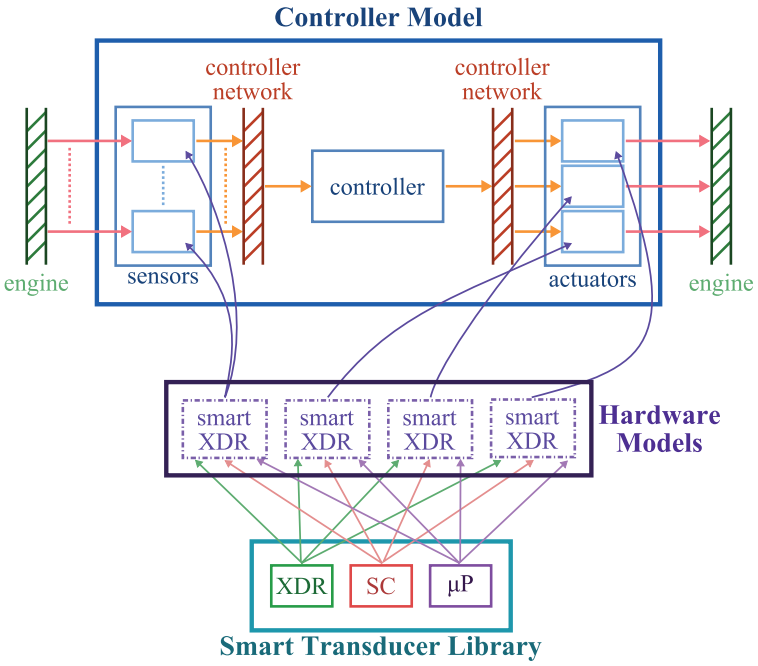

Figure 5. Graphical representation of the modular framework for modeling the CM. Hardware models are constructed using transducer (XDR), signal conditioning (SC), and signal processing (μP) components from the Smart Transducer library and integrated in the CM, which interfaces with the EM and WM (not shown).

One final drawback of this modeling approach is that the smart transducer and controller network models presently available, and used for the simulations presented in Section IV, are unvalidated due to the absence of experimental data demonstrating the functionality of the components of the CM. This limits the use of results from simulation of the CM to proof-of-concept until more information is made available about the characteristics of the smart transducers and network expected to be used in a distributed control implementation. Like the limitations on model fidelity related to the computational cost of simulation, this drawback may be addressed as distributed controls technology advances. The benefits of this modular approach, developed to encourage collaboration and provide flexibility in the modeling of a distributed control architecture, far outweigh the drawbacks, many of which will become less substantial as DEC technology advances.

IV. Demonstrating the Smart Transducer Library

The Smart Transducer Library, introduced in Section III, is organized as illustrated in Fig. 4. The "Transducers" sublibrary presently contains relatively low-fidelity models of sensors and actuators. The functionality of blocks in the "Signal Conditioning" sublibrary is limited to basic tasks such as filtering, compensation, and conversion, and the "Application Processor" sublibrary contains blocks for averaging, unit conversion or linear mapping, extrapolation, and local loop closure. The only block presently in the "Network Connection" sublibrary models the (random) delay and packet loss possible as data are transferred on a network cable. More accurate models and expanded processing capability can be added as hardware specifications and processing needs become better defined.

Low-fidelity hardware models may be constructed by combining blocks from the library in a manner similar to the diagram in Fig. 3. For a sensor, the hardware model is connected to a filter, compensator, analog-to-digital converter, and one or more processing blocks to form the 'smart sensor' model. An actuator is constructed analogously, with local loop closure using data fed back from a smart sensor that may be integrated with the smart actuator or connected over the network. The output of the sensor (or the controller) may be directly connected to the controller (or actuator), or may be routed first through a block modeling the network. Once integrated in the CM, the smart transducer models may be upgraded incrementally, by replacing single blocks at a time, or completely, by replacing the entire hardware model with a new configuration of blocks.

To demonstrate how the data processing functionality in models constructed from the Smart Transducer Library can affect simulation results, the C-MAPSS40k engine was connected in closed-loop with three controller models: the baseline model discussed in Section II (the 'unstructured' model) and two models with the distributed architecture discussed in Section III, one without and one with a model of the controller network (the 'distributed' and the 'networked' models, respectively).

A. Implementing a distributed architecture model

Transitioning the unstructured controller model to one with a distributed hardware architecture required two main changes: upgrading the sensor and actuator models to ones constructed using the Smart Transducer Library, and introducing the ability to assign update rates individually to each component. The networked controller model also captured the effects of communicating the outputs of each feedback sensor and each command output of the controller over the network. The sensor models were a cascade of blocks modeling the hardware, filter and other compensation, and the quantization of the measurement by the analog-to-digital conversion (ADC). Similarly, the actuator models contained filters representing conversion and conditioning of the quantized control command and a model of the hardware dynamics. Two actuators required feedback sensors to provide data for locally closing the loop to ensure adequate tracking of the control command. For each smart transducer, characteristics of the transducer hardware, signal converter, and data processing tasks must be specified. The information listed in Table 1, for a smart pressure sensor, prescribes the input and output range of the sensor (in pounds-force per square inch, psi, and voltage, V, respectively), the response time (in seconds, s) of the sensor output, the input voltage range and resolution (in bits) of the ADC, and the size of the averaging window used to process the data. This information is representative of what may be found in the TEDS. The network model block was configured with the time delay distribution and packet drop likelihood also provided in Table 1. Note this network configuration is an exaggeration from what is expected in order to better demonstrate the effects of the network model.

Further realism was added by allowing for multiple update rates, T_s, within the CM. In the unstructured controller model, each component is simulated with a common clock incrementing by a fixed time-step (the controller update rate). In reality, the individual components (smart sensors, smart actuators, and the controller) operate asynchronously using their own clock. For simplicity, the distributed and networked controller simulations were configured to run at a fixed time-step equal to the sensor update rate (the same for all sensors), while the controller and actuators each operated at different multiples of this time-step. This introduces realism to simulation without the increased complexity required to implement asynchronous clocks as characteristic of a physical control system. By ensuring that update rate transitions were properly handled, modeling a controller network that keeps only the latest data from each node, the CM could be visualized as a collection of functions accessing the network at different rates, as illustrated in Fig. 6.

B. Comparison of models

Simulations of the closed-loop engine with each controller model were run to demonstrate how the added complexity in the distributed and networked models manifests itself in the simulation results and in the simulation run time. The latter is important to consider because real-time operation is necessary when one or more hardware prototypes are to be simulated in the loop with computational hardware models. Results

Table 1. Sample configuration data used for a smart pressure sensor model and network block. The smart pressure sensor places on the network the average of a specified number of scaled voltage (V) measurements made by the on-chip sensor. The measurements are proportional to input pressure (in pound-force per square inch, psi) and mapped to the input voltage range of the ADC. The network model imposes a random packet delay (in seconds, s), with the possibility of packet loss occurring at a specified rate.

Sensor model configuration		**Network cable model configuration**	
Sensor input range (psi)	0 to 30	Average delay (s)	0.001
Sensor output range (V)	0 to 0.07	Delay standard deviation (s)	0.003
Sensor rise time (s)	0.0879	Packet-drop probability (%)	15
ADC range (V)	−5 to 5		
ADC resolution (bits)	8		
Averaging window (sample)	3		

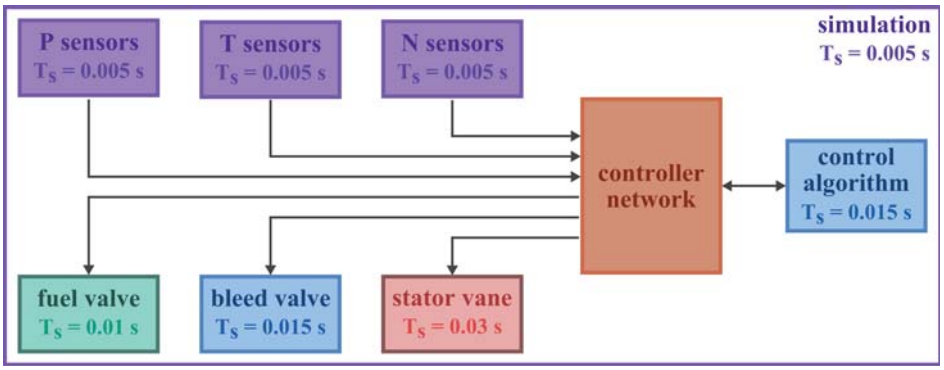

Figure 6. Components of the distributed and networked controller models, color-coded by update rate T_s.

are presented for simulation of the models given a throttle input commanding a series of small changes over 60 seconds. The performance of each controller was evaluated by comparing the tracking of the engine pressure ratio and the thrust produced. From Fig. 7, it can be seen that the general performance of the controller is not significantly affected by increased hardware model complexity: the control variable is tracked to relatively the same degree of accuracy by each controller model.

The most noticeable difference between the results are small oscillations, and a slight negative off-set, particularly observable when the throttle command is constant for an extended period of time, as seen in the inset plot in Fig. 7. These can be attributed to the quantization of the feedback signals by the ADC in the smart sensors. This is highlighted in the bottom plot of Fig. 8, where the input and output of the low-pressure turbine exit pressure sensor (P_{50}) are compared for the three simulations. Here, the offset (due to the flooring operation of the ADC) and oscillation between quantization levels by the sensor output signal is clearly seen. Because this quantized signal is fed back to the controller and the reference signal that is being tracked is in between quantization levels, the controller responds by commanding an input that correspondingly oscillates as it tries to correct for the tracking error, as can be seen in the top plot in Fig. 8. The amplitude of this oscillation is within the quantization error of the ADC, so an increase in resolution (and adding filters to the actuator command) may reduce these numeric oscillations.

Despite the exaggerated network cable model configuration, the effect of including simple network models is barely noticeable in Fig. 8. In general, the network model imposes random delay between when data are

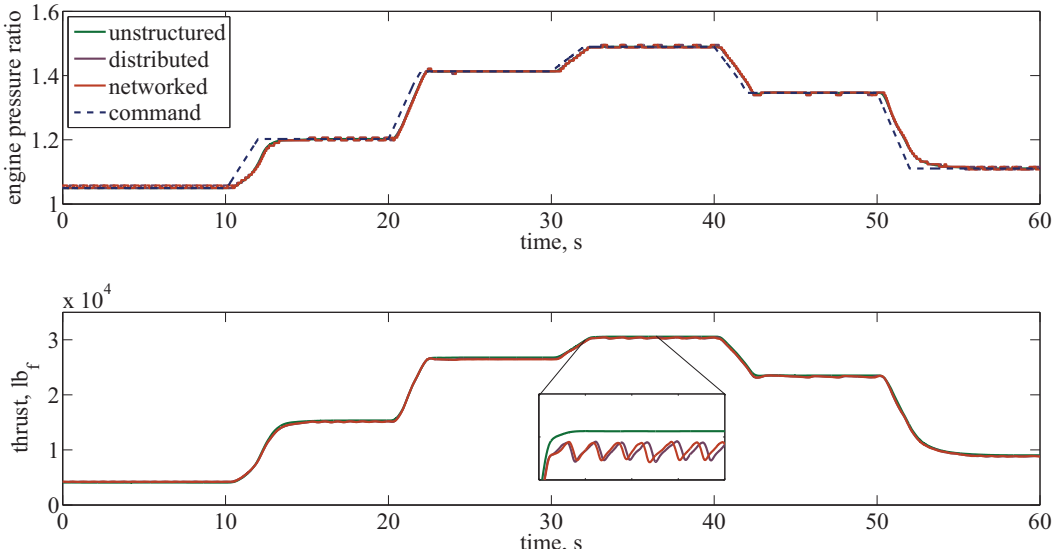

Figure 7. Tracking of the engine pressure ratio and thrust production in simulation of the C-MAPSS40k engine with three different controller models. The inset graph provides a closer view of the results.

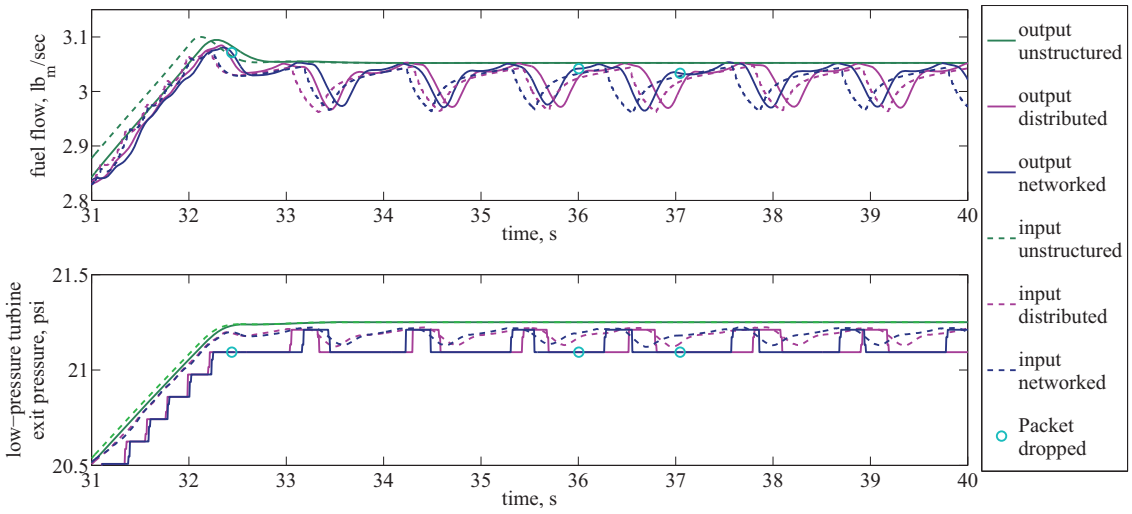

Figure 8. Inputs (dashed lines) and outputs (solid lines) of the fuel flow actuator and P_{50} pressure sensor during simulation of the C-MAPSS40k engine model with three different controller models. The sensor input is from the engine, while the actuator receives a command from the controller; the outputs go to the control algorithm and engine, respectively.

written to the network and when they are available to be read. This is most clear in the sensor response from 31 to 32 seconds, where a small (and varying) lag is present between the response in the distributed and networked models as the engine accelerates. The block also models random data loss, when packets are not delivered in time to be read from the network; this is indicated by circles in Fig. 8. Extrapolation was implemented to account for these losses, resulting in little effect on the results. Although used here to recover lost data, this library block may also be adapted for use in diagnosing malfunctioning sensors by checking data consistency.

To verify that real-time simulation of the distributed and networked models is possible, 200 simulations of each model (with the same input) were timed using built-in MATLAB commands. The box plots in Fig. 9 capture the distribution of run times for each model. The variations, measured as the standard deviation of the data, were relatively small (0.8111, 0.8785, and 1.1982 seconds, respectively, for the unstructured, distributed, and networked models) and likely due to varying demands on the computer processor and memory during each simulation. The extra demand required for the more complex distributed and networked controller models was most evident in the increase in average run time from 5.5403 seconds to 26.1845 seconds and 34.097 seconds, respectively. Despite being significantly higher, these run times were 3.06 and 2.35 times faster than real-time, respectively, suggesting that simulation with one or more computation hardware models replaced by actual hardware should be possible.[a]

V. Summary

Centralized and distributed engine control differ in how the physical control system is implemented. Rather than including data processing functionality in the ECU, as done in a centralized control scheme, these tasks are embedded within sensor and actuator hardware to form the smart transducers used in a distributed control architecture. This allows for digital signals to replace analog signals for communication between the transducer hardware and the ECU. A computational model of the control platform used in an HIL setting must capture the effects of this hardware arrangement in order to enable simulation with and without physical hardware in the control loop. Here, a modeling framework for the control system has been proposed to introduce modularity to the model through the establishment of well-defined interfaces between the controller and the hardware models.

This framework extends to the hardware model level and the Simulink library being created to provide

[a]Although these results suggest that, on average, real-time simulation of these models is possible, additional investigation is needed to verify that the real-time constraint is not violated during time-steps when the engine takes longer to converge (such as during an input transient).

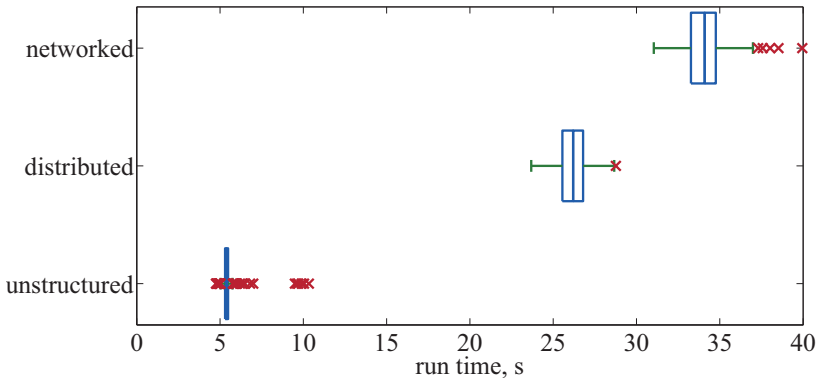

Figure 9. Run-times for 200 80-second simulations of the C-MAPSS40k engine model with each of three different controller models. Note that even the slowest-running model still runs faster than real-time.

models of the components in an IEEE 1451-compliant smart transducer. The library contains blocks for modeling actuator and sensor hardware, signal conditioning and conversion functions, data processing operations, and a network cable. Blocks from this library may be combined to form models of smart sensors or smart actuators to be incorporated in the model of the controller platform. Here, the C-MAPSS40k engine model was simulated in closed-loop with three controller models: the C-MAPSS40k controller, which contains simple sensor and actuator models with no data processing effects, and two controllers modeling smart transducers. The results show that the overall tracking ability of the controller is not significantly affected by the more complex hardware models, while previously unmodeled effects related to the limited resolution of the ADC in the smart sensors can be observed. Although more computationally intensive, the models with distributed architectures can still be simulated faster than real time, as required for models intended to be simulated with hardware in the loop.

By requiring consistently-defined interfaces at each level of this framework, the model takes on a modular structure. The ease with which components can be removed and replaced allows for collaborative development, and aligns with the main goal of the HIL system: providing a means to test control architecture or hardware in a low-risk setting. Preliminary testing requires a computational model of an engine, and specifications (or models) for the controller architecture and hardware that fit within the modeling framework.

Possible follow-up investigations involve replacing hardware models with physical prototypes and using a physical controller network to develop better computational models of the network. This will enable verification of the accuracy of the computational controller and hardware models in predicting the actual performance of the hardware.

References

[1]Culley, D. E., Thomas, R., and Saus, J., "Concepts for Distributed Engine Control," *Proceedings of the 43rd Joint Propulsion Conference and Exhibit*, AIAA-2007-5709, Cincinnati, OH, July 2007.

[2]Culley, D., "Transition in Gas Turbine Control System Architecture: Modular, Distributed, and Embedded," *ASME Turbo Expo2010: Power for Land, Sea, and Air*, Vol. 3, Glasgow, Scotland, United Kingdom, June 2010, pp. 287–297.

[3]Yedavalli, R., Belapurkar, R., and Behbahani, A., "Stability Analysis of Distributed Engine Control Systems Under Communication Packet Drop," *Proceedings of the 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2011-6145, Hartford, CT, July 2008.

[4]Belapurkar, R., Yedavalli, R., and Moslehi, B., "Stability of Fiber Optic Networked Decentralized Distributed Engine Control Under Time Delays," *Proceedings of the 45th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2009-4885, Denver, CO, August 2009.

[5]Yedavalli, R. K. and Belapurkar, R. K., "Design of Distributed Engine Control Systems for Stability Under Communication Packet Dropouts," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 5, September-October 2009, pp. 1544–1549.

[6]Belapurkar, R., Yedavalli, R., and Behbahani, A., "Study of Model-based Fault Detection of Distributed Aircraft Engine Control Systems with Transmission Delays," *Proceedings of the 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2011-5795, San Diego, CA, July 2011.

[7]Yedavalli, R., Willett, M., and Behbahani, A., "The Role of Various Real-time Communication Data Bus for Open System Distributed Engine Control Architectures for the Future," *Proceedings of the 47th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2011-6145, San Diego, CA, July 2011.

[8]Culley, D., Thomas, R., and Saus, J., "Integrated tools for future distributed engine control technologies," *Proceedings of the ASME Turbo Expo 2013*, GT2013-95118, San Antonio, TX, USA, June 2013.

[9]Culley, D., Zinnecker, A., and Aretskin-Hariton, E., "Developing an Integration Infrastructure for Distributed Engine Control Technologies," *Accepted to AIAA Propulsion and Energy Forum and Exposition 2014: 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, Cleveland, OH, July 2014.

[10]Aretskin-Hariton, E. D., Zinnecker, A. M., and Culley, D. E., "Extending the Capabilities of Closed-Loop Engine Simulation using LAN Communication," *Accepted to AIAA Propulsion and Energy Forum and Exposition 2014: 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, Cleveland, OH, July 2014.

[11]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats," September 2007.

[12]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Network Capable Application Processor (NCAP) Information Model," 2000.

[13]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," 1998.

[14]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems," April 2004.

[15]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," 2004.

[16]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," October 2007.

[17]"IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducers to Radial Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats," June 2010.