Validation of Hardware and Software in the Loop add-ons simulation modules

Mohamad Haffar, Jean-Marc Thiriet, Sarah Kabbara

Validation of Hardware and Software in the Loop add-ons simulation modules

Mohamad Haffar[1, 2], Jean Marc Thiriet[2], Sarah Kabbara[2, 3]

[1] Euro System, 26 chemin de la digue, BP 28, 38760 Varces - France
[2] GIPSA-lab, 961 rue de la Houille Blanche, BP 46, 66121 Saint Martin d'Hères cedex - France
[3] Université Libanaise Faculté de Génie Branche 1, Kobbé, Tripoli - Liban

mohamad.haffar@euro-system.fr jean-marc.thiriet@ujf-grenoble.fr hs3y@hotmail.com

***Abstract* – The integration of communication networks in critical or safety environment such as Substation Automation System "SAS" incorporating supervisory control and data acquisition system "SCADA" is nowadays a strategic research problem. New standards of communication such as the IEC 61850 appears, in order to solve critical situation, ensure the security and avoid the blackout of the whole system. However, the reliability of the system relies on sending a large number of real time messages via the communication network which increase the usage of the network bandwidth. Our approach consists of setting up a new platform for simulating a SAS network in order to analyze the various information flows which share the network, check that the temporal constraints. Modeling and validating of network components is an essential step before running any simulation. In this paper we will present the concept of modeling an IED communication server and the usage of Hardware and Software In the loop modules for validating the implemented protocol functionalities.**

***Key Word* – Co Simulation Platform, hardware in the loop, Software in the loop, protocol functionalities, Modeling, OpNet Modeler, SCADA.**

1. List of abbreviation

The following table 1 contains the definition of acronyms used in this paper.

Acronym	Definition
IED	Intelligent Electronic Device
SAS	Substation Automation System
FAT	Factory Acceptance Test
HITL	Hardware In The Loop
SITL	Software In The Loop
GOOSE	Generic Object Oriented Substation Event
DPA	Distributed Protection Application
ODBOC	OpNet DeBugger Output Consol

Table 1: Abbreviation List

2. Introduction

A substation automation system 'SAS' is used for power distribution whatever is the source of electricity production. Within a SAS, all kind of high power equipment such as switches, transformers and high power lines are used to ensure the power distribution from the higher to the lower voltage level. The protection of these equipments is provided by an intelligent electronic device "IED" [1].
Two type of protection exists to ensure the overall security of the SAS architecture.

Local protection is used inside a primary protection device and aims to guarantee the security of the electrical devices linked to the primary equipment.
DPA is the second type of protection. It is based on an exchange of automation information between primary devices. It aims to ensure the reliability of the overall SAS architecture. The sharing of information between devices was previously accomplished using hardwired cable links. Nowadays, with the appearance of new standards, DPA becomes attainable through the communication network. This point has a big interest in the world of substation since it permits to reduce the hardwired notion and provides a single media for communicating all the information between all SAS components.

However, when this feature is used, the evaluation of communication network performance becomes an essential task to satisfy during SAS project development. Until this time, the overall performance and extensibility of the SAS communication network are still left unanswered. Many approaches (e.g. simulation approach [2], analytical approach [3] and experimental approach [4], [5]) exist in order to accomplish this task. However all these approach cannot be used in SAS performance evaluation for many reasons explained in [6]. Therefore, in our research, we developed add-ons simulation module in order to construct a Co-Simulation platform. This platform aims to provide a common methodology for SAS project phases.

Any primary device could be logically divided into two sections; the functional section (i.e. protection and control) and the communication section [7]. Modeling of communication section inside the simulation software is an essential task during the construction of platform. To increase the confidence in our Co-Simulation approach, each time a model is developed, our study consists of testing and validating the communication functionalities implemented inside the model according to the specification of the communication protocol.

In this paper we will represent the modeling of a communication server inside the OpNet simulation software and the usage of add-ons modules developed in our research for validating the model.

3. Co Simulation Platform Construction

The details of construction of our Co-Simulation module can be found in the [8]. To give a small outline, our Co-Simulation platform can be divided into three functional modules as shown in the Fig. 1:

- Simulation software
- Software in the loop package
- Hardware in the Loop package

The simulation software is software in which all the network devices will be modeled and in order to construct the network architecture and evaluate its performance.

Hardware In the loop module is package that permits the connection between virtual models developed inside the simulation software and real device outside the simulation world.

Software in the loop package is another module that permits any external application or consol to connect to the simulation software in order to exchange information with the process of the modeled device at simulation runtime. Therefore the behavior of these devices can be dynamically changed using this module.

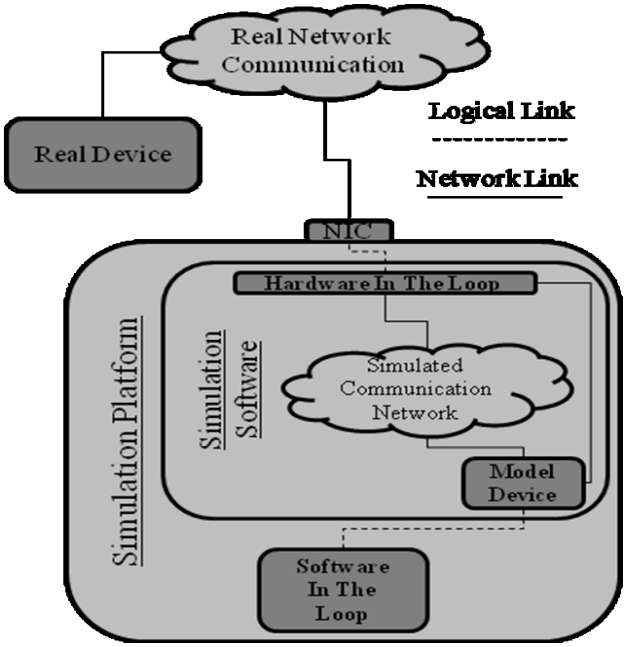

Figure 1: Co-Simulation Platform Architecture

4. Background of OpNet Modeler Simulation Software

Each model inside the simulation contains many modules connected with a packet stream. The implementation of the communication protocol takes place at the application module of the device model. The other modules represent the modeling of the OSI layers (e.g. TCP/IP communication stack). Standard protocols for lower layers such as TCP and UDP for the transport layer, IP for the network layer, Ethernet for the data link layer are provided by the simulation software inside its models library. Nevertheless, due to the diversity of the industrial functionalities and protocols at the TCP/IP application layer, simulation library doesn't implement industrial protocols application. Therefore, modeling of device should be achieved by OpNet developers by implementing application protocols at the highest module of the modeled virtual device.

Each layer is called module in OpNet simulation language, and each module is associated to a number of attributes that defines its behavior. More particularly clients and servers own many attributes, two of them are crucial to define before composing any simulation architecture:

- The process attribute
- The compound connection attributes.

The process attribute contains the name of the process model associated to the application module; therefore this attribute must point to the process of the programmed application. The compound connection attribute indicates the number and types of connection the model is configured to initiate. Passive connection command are send from the application to the transport layer when the model is configured as a server node and active command are sent when it is configured as a client node. The process model of the virtual device is programmed in a manner to accept a double work (client and server in the same time) for further simulation tests.

Three essential editors constitute the simulation software as shown in the Fig. 2. The project is the editor in which the network architecture is constructed. The node editor represents the

communication layers inside each node model. Finally the process of each module is constructed in the process editor.

The process model attribute of the application contains the program of this module and is developed in a finite state machine 'FSM' object programming language which facilitates the implementation of application and algorithm. States and transitions define the progression of a process in response to events. Events are generated in our study via the software in the loop package and affect the model behavior in simulation runtime mode. Two different types of states exist in this software the forced and unforced states. Each of them have an enter executive and exit executive part that contains embedded C/C++ code. The major difference between a forced and an unforced state is that when a process executes an unforced state, it completes the enter executive of the state and blocks by returning control to the previous context that invoked it. The unblocking procedure is normally done by reviving the process execution after reception of an internal or external module interruption. However a forced state does not allow a block between its enter and exit executive and once it is invoked, all the code contained inside its two parts is executed.

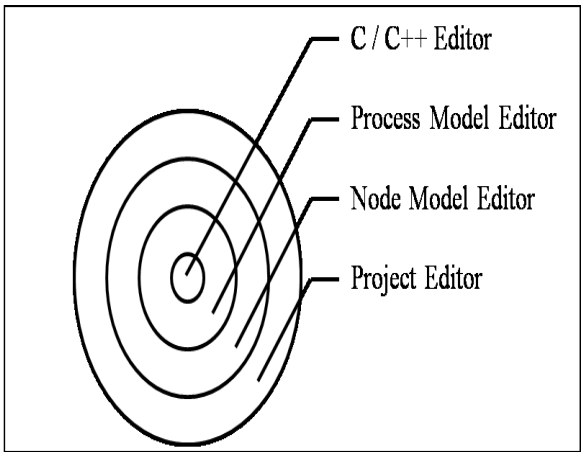

Figure 2: OpNet Simulation editors

5. Modeling of IED server

In this paper we will present the modeling of an IED device acting as a communication server. In order to validate the implementation of HITL and SITL as well as the modeling of the device, we choose a well known industrial communication protocol 'ModBus IP' for many reasons:

- Possession of a complete knowledge about this industrial protocol being an open source protocol which makes easier the modeling task.

- The widespread usage of this protocol and the diversity of industrial client/server equipments which is helpful in models validation phase and allows the comparison of network performance between simulation results and real experimental outcomes.

- Existence of this protocol in a Master/Slave mode which helps the validation of simulation results by comparing to analytical formula.

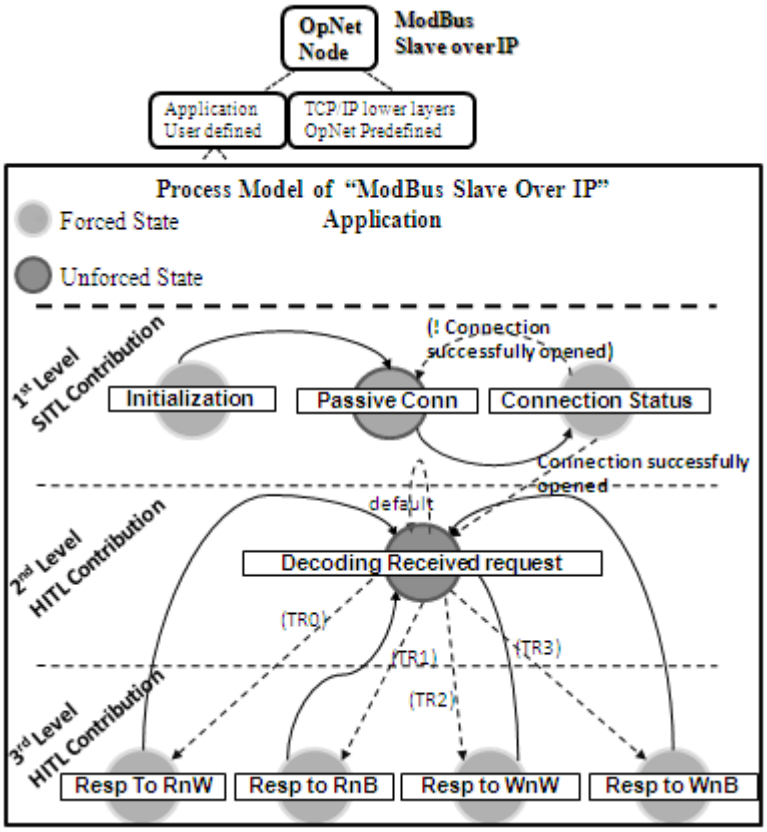

Figure 3: IED server Process model

The process model of this network model is shown in the Fig 3. This process is divided into three different levels.

The process initialization and the communication connection establishment are accomplished at the **1st** level of this process. In **'Passive_Con'** state, the preconfigured combined attribute are parsed inside the process of the model on simulation runtime in order to initiate an Open passive connection according to the IP address and port number defined inside the compound attributes.

An IED server normally contains an internal data memory that can be accessed from SCADA systems. For ModBus this data memory is divided into two areas:

- Bit memory
- Word memory

SITL module intervenes at this level in order to configure the data memory of the modeled device. In fact at the **'Initialization'** state, the model performs a connection to the shared memory of the SITL pack and recuperates all the initialization bits and words. In this case the shared can be initialized in two columns the bit columns and the word columns.

As shown in the Fig. 3**, 'Passive Con'** is designed to be an unforced state. Hence, once the process enters this state it blocks the control until the receiving of a simulation event. The event will be treated in the **'Connection Status'** state. If it corresponds to connection establishment (i.e. a Synchronization requests coming from a distant client) then the model acknowledges the connection demand and transits to the next level else the process return to the state and wait for an active connection.

The **2nd** level contains just one unforced state. In this level the model blocks the process and waits for an event indicating the reception of a communication request. HITL package intervenes in this state to bind to Network interface card of the model to the network interface card of the simulation machine. Therefore, any remote request sent to the destination IP of the model are automatically routed by the HITL router edge from the NIC of the simulation machine to NIC of the simulation model. Once a request is received, the state **'Decoding Received Request'** checks the type of the received request by exploring requests fields.

Four ModBus functionalities are implemented inside the model. The Read Bits Read Words, Write Bits and Write Words. Depending on the type of request decoded in the 2nd level, the process activate to corresponding transition for constructing the response. This mission is done at **3rd** level of the process model. HITL intervenes also inside this level in order to send the constructed response to the real client outside the simulation.

On final the process returns to the **2nd** level and waits for another communication request.

6. Co-Simulation architecture and model validation results

Experimental setup

In order to validate the implementation of ModBus Functionalities inside the modeled server, an experimental setup based on the simulation model, real zone, SITL package and the HITL package is constructed in this study.

The real zone corresponds to the real components that will be connected to the simulation model. In our case, an industrial SCADA system based on the ModBus protocol is chosen in order to validate the communication functionalities of the server. This component is named **'Vijeo Designer'**.

HITL will serve to connect the IED server model to the SCADA System. SITL will serve to configure the internal data memory and the characteristics of the simulation model.

The Fig. 4 shows the logical links, implementation links and network links between the different zones of the experimental set up. A logical link exists between the external consol and the shared memory as well as between the shared the memory and the model server for exchanging information between the external application and the OpNet models on simulation runtime. Another logical link exists between the HITL router edge and the NIC of the simulation machine in order to ensure the binding of the real IP address of the simulation machine to the IP address of the simulation model. Finally an implementation link exists between the HITL Lib and the HITL router in order to indicate that the HITL pack is not capable to work without the implementation of the HITL Lib developed in our study.

All this experimental setup and its internal links aims to provide at final a network connection between the real device inside the real zone and the server model inside the simulation software.

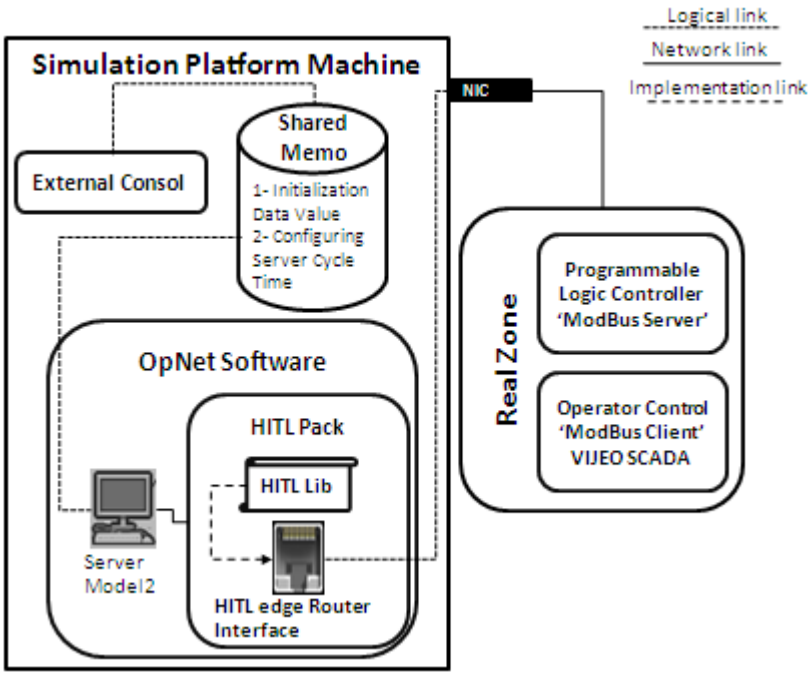

Figure 4 : Experimental setup

SCADA application

A small SCADA application is developed in this study in order to validate the HITL, SITL operation as well as the modeling of server. For this end, two separated zone are introduced in this application.

The first one, on the left of the Fig. 5 contains two write commands one of them is for the bit data memory and the second for the word data memory. The two commands are configured to change the word and bit located at the address 5 of the server data memory. This zone permits the validation of the ModBus write functionalities implemented inside the model.

The second zone, on the right of the Fig. 5 contains four read commands. These commands are divided as follow:

- Two commands for reading bit memory at address 4 and 5
- Two other commands for reading word memory at address 5 and 6.

The reason behind this division is to retrieve the value written by the client after sending the write bit/word requests at address 5.

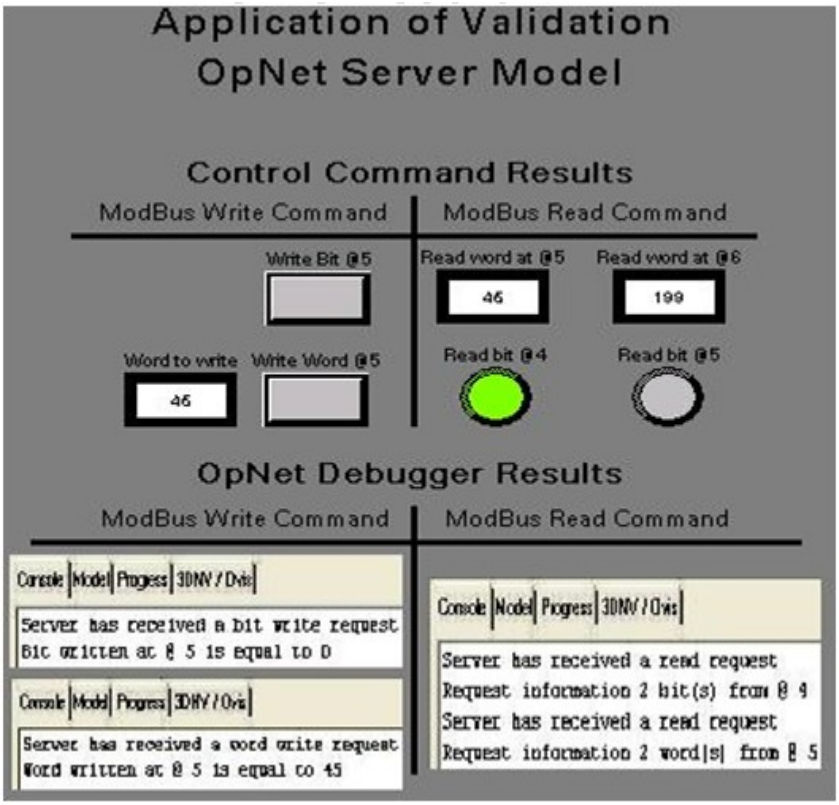

Figure 5: SCADA application

Validation Results

Validation results can be shown in two different methods:

- At the server side using 'ODBOC'.
- At the client side using the SCADA application

These two types of validation are provided in our study in order to ensure a high level of confidence for our validation concept.

When it comes to the validation of Write Functionalities, normally there will be no value returned for the real client, hence, ODBOC indicates the reception of the remote write requests and indicates also the changes made inside the model data memory. This functionality can be also confirmed at the client side by sending a read command after the write request and ensuring that the data received is equal to value that was written.

However the read functionality is normally treated at the remote client side, in fact to confirm this functionality, the client must display the values of the words and bits found inside the server.

SITL was configured to initialize all the word data memory to the 199 value. All bits inside the bit data memory are at its turn initialized as active '1'.

At the running of the SCADA application, the remote client generates automatically and periodically read requests to the configured server model. The write requests are sent on human demand. Therefore as shown in Fig. 5, the 199 value will displayed inside the two fields of the read word request until the reception of the remote write request changing the value of the word from 199 to 45.
The same scenarios will happen to bit data memory.

7. Conclusion

This demonstration papers explain the validation of a Co-Simulation Platform. HITL and SITL are the two essential components that are successfully developed in our study. These modules are henceforth operational in our study. As explained in this paper, these module are very helpful for constructing the platform since they provides a methodology for validating communication models. Many other interests are hidden behind these module, they can ensure a way for reducing the simulation time and realizing a network simulation composed by real devices and simulation models.

8. References

[1] C. Strauss, Practical Electrical Network Automation and Communication Systems*, ISBN: 0750658010, Ed. Great Britain: Newnes, 2003.*

[2] Pereira N., Tovar E., Pinho L. M. (2004), INDEPTH: Timeliness Assessment of Ethernet/IP-based Systems, MASCOTS 2004, Volendam Netherlands, 192-201.

[3] Robert J., Georges J-P., Rondeau E., Divoux T. (2010), Analyse de performances de protocoles temps-réel basés sur Ethernet, CIFA, Nancy France.

[4] Kalappa N., Acton K., Antolovic M., Mantri S., Parrott J., Luntz J. , Moyne J., Tilbury D. (2006*), Experimental Determination of Real Time Peer to Peer Communication Characteristics of EtherNet/IP,* ETFA'06, Prague Czech republic, 1061-1064.

[5] Alessendria E., Seno L., Vitturi S. (2007), *Performance Analysis of Ethernet/Ip Networks*, FET, Toulouse France, 391-398.

[6] Haffar M., Thiriet J-M, Kabbara S. (2010), *A Hardware in the Loop module in an IEC61850 Co- Simulation Platform for advanced substation automation system tests*, IEEE Energycon, Manama Bahrain.

[7] Bruandet G., Angays P, Haffar M., Saxena P. (2010), *IEC 61850 Tutorial*, PCIC, Oslo Norway.

[8] Haffar M., Thiriet J-M., El-Nachar M. (2010), *Software and hardware in the loop component for an IEC 61850 Co- Simulation platform*, IMCSIT, Wisla Poland.