Proceedings of the 7th WSEAS International Conference on Automation & Information, Cavtat, Croatia, June 13-15, 2006 (pp53-61)

FLEXICON Development Process with Co-simulation for a Marine Application

D.N. RAMOS-HERNANDEZ, H. A. THOMPSON, J. FU, L. JIANG, J. NIU AND D. DOBINSON[1]
Department of Automatic Control and Systems Engineering
University of Sheffield
Mappin Street, Sheffield S1 3JD
[1]Marine Electrical Systems
Rolls-Royce plc
UNITED KINGDOM

Abstract: Developing distributed control systems requires tools that can deal with high complexity, integrating design information from multiple domains and allowing analysis of the overall design in terms of technical and commercial performance. Currently, no integrated multidisciplinary design tools exist which can do this. Thus, the ability to simulate and assess system performance and rapidly prototype systems is critical. The FLEXICON project aimed to solve these issues and for this a toolset for Marine, Automotive and Aerospace (MAA) applications was created. This paper presents a way to reduce the development process time in the production of a marine application using the MAA toolset and the concept of co-simulation adopted in the FLEXICON project by the University of Sheffield. A reduction of 30% in development was achieved for the Marine application. Information on these results was obtained during the Marine demonstrators activities.

Key-Words: Co-simulation, distributed systems, reliability & through-life-cost, development process, modelling.

1 Introduction

Adopting distributed systems rather than centralised ones is a great concern for Industry since this change involves cost and risks for their businesses. Developing a distributed control system requires tools that can deal with high complexity, integrating design information from multiple domains and allowing analysis of the overall design in terms of technical and commercial performance. Currently, no integrated multidisciplinary design tools exist which can do this. Thus, the ability to simulate and assess system performance and rapidly prototype systems is critical.

The FLEXICON project (IST-2001-37269) funded by the EU aimed to reduce the development life-cycle and implementation cost for distributed control systems. Thus, a toolset for Marine, Aerospace and Automotive applications (MAA) was implemented within the project. This toolset allows the integration of Commercial Off-The-Shelf (COTS) tools (via co-simulation and with hardware-in-the-loop) for the design and development of these applications. The COTS tools integrated are ISaGRAF Enhanced™, Simulink™/Stateflow™, ARTISAN RtS™and Excel™. Also, the toolset allows analysis of control system performance, fault tolerance availability and through-life-cost. FLEXICON uses open standards such as OPC, IEEE 1451 and other standards like UML, IEC 61131-3, CORBA and NDDS [1].

In order to demonstrate the flexibility and scalability of the MAA toolset, a portable and full demonstrator for a Marine application were built. The Marine application was of a waterjet driven high speed vessel with five propulsion systems as described in [2] and Section 3 of this paper.

To demonstrate the steering and reversing capability of the waterjets a scale model boat was used. The demonstrators use mainly Pentium 4 processors for the integrated COTS tools and also for the ***MAA tools*** developed (Co-simulation, Reliability & Through-Life-Cost, Configuration Management Database, Java Code Generator, VHDL Code Generator, System Builder, tools support for Co-simulation with Hardware-In-the-Loop, and Alarms and Warning Management). Also, PowerPC, Tecla PLC and FPGA processors are part of the demonstrators. Analog devices and network such as Ethernet, CAN, Wireless Bluetooth and Serial connections are also included.

2 FLEXICON MAA toolset

The objective of the FLEXICON project was to construct a toolset from COTS tools to reduce the development life-cycle and cost of distributed control applications. During the project several tools were developed which constitute the MAA toolset. As part of the toolset a co-simulation environment was constructed from different COTS tools. Also, the co-simulation allows the integration of fault tolerant systems via co-simulation with hardware-in-the-loop. In addition, availability and through-life-cost of the applications is calculated automatically using this environment.

Fig. 1 shows the development process model adopted in the FLEXICON project. This model was the result of several discussions and comparison of the development processes of the FLEXICON End User partners. This diagram also corresponds to the V-model. For the MAA toolset, as shown in the figure, co-simulation is very important for requirements capture, performance analysis and costing, and also during the testing and commissioning phases. The MAA toolset also supports the code generation and the aftermarket phases.

The MAA toolset will help in the development process of a new application. Conceptual requirements provided by the end-user and analysis of these requirements are initially implemented using the MAA co-simulation environment. Also, an initial study of Reliability and Through-Life-Cost (R&TLC) is performed. This provides feedback to the end-user and other requirements can also be considered. Then, a detailed design of the system is carried out via detailed modelling and co-simulation. After finishing the design phase, coding using the MAA code generators and COTS code generators is performed. Testing using co-simulation with Hardware-In-the-Loop is executed, checking again the design and analysis. If everything is okay, then commissioning of the system with co-simulation support and Hardware-In-the-Loop can take place. If the system integrators are satisfied with the tests then the system is passed to the end-user to carry out acceptance testing. The end-user will test the system against requirements. The R&TLC can be recalculated using the latest data.

Following this development process, the Marine application was implemented. A general description of the application is given in Section 3.

2.1 Co-simulation in the Specification and Design Phase

Co-simulation is used in the specification and design phase and also in the specification and design verification phase.

In the Specification and Design Phase, the control system is taken from concept design to final design. Initially, specifications for the hardware and software designs are gathered. Then candidate designs are evaluated and analysed allowing the control system engineer to narrow down the possible designs to one that will be taken forward. The final hardware and software designs are then completed.

During the Specification and Design Verification Phase, the hardware and software designs are evaluated and analysed to confirm that they meet the technical requirements for the product.

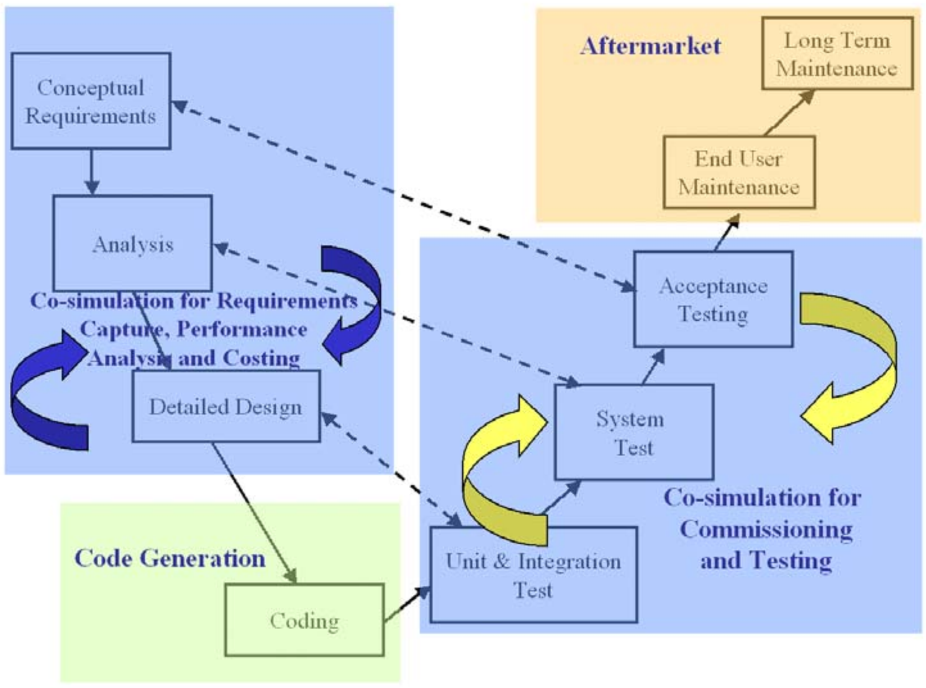


Fig. 1: FLEXICON Development Process Model with Co-simulation

Fig. 2 shows how the co-simulation can be used during these phases. In this example, the COTS tools integrated via co-simulation are:

- ISaGRAF Enhanced has been used for discrete control system modelling
- Simulink has been used as a continuous system model simulation tool
- ARTISAN RtS has been used as a model building and management tool
- Microsoft Excel has been used for reliability and through-life-cost calculation

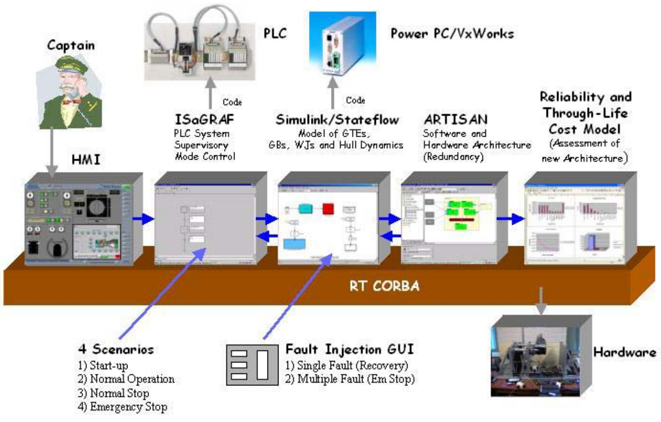
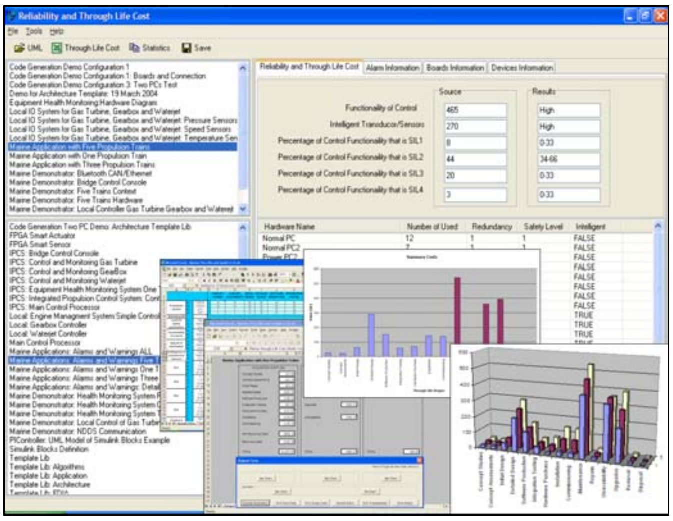

Captain
HMI
PLC
Code
ISaGRAF
PLC System Supervisory Mode Control
Power PC/VxWorks
Code
Simulink/Stateflow
Model of GTEs, GBs, WJs and Hull Dynamics
ARTISAN
Software and Hardware Architecture (Redundancy)
Reliability and Through-Life Cost Model
(Assessment of new Architecture)
RT CORBA
4 Scenarios
1) Start-up
2) Normal Operation
3) Normal Stop
4) Emergency Stop
Fault Injection GUI
1) Single Fault (Recovery)
2) Multiple Fault (Em Stop)
Hardware

Fig. 2: Co-simulation vision

In combination, the ISaGRAF and Simulink tools can be used to rapidly prototype the control system by testing discrete and continuous laws developed in ISaGRAF and Simulink against a model of the plant in Simulink. UML modelling (with the ARTISAN RtS tool) is used to define the hardware architecture redundancy for fault tolerance and the functionality of the control system. Fault injection scenarios can be injected into Simulink to test out fault detection, isolation and accommodation strategies. Excel is used to perform reliability and through-life cost modelling of the system. At a later stage co-simulation with Hardware-In-the-Loop is possible allowing integration testing of real system components within the prototyping environment.

In order to calculate automatically the reliability and through-life-cost of several hardware system designs within the co-simulation environment, the **Reliability &Through-Life-Cost (R&TLC) tool** is used. This tool allows calculation of system reliability figures through extraction of data from the design tools. Also, this tool calculates automatically the cost of the system as the system's architecture is defined in UML and control functionality is defined in Simulink and ISaGRAF. For instance, information on Safety Integrity Levels (SIL) and sensor types (intelligent and dumb) built into the UML model is extracted. Fig. 3 shows an example of the reliability and through-life-cost calculation for different hardware option designs of a system.



Code Generation Demo Configuration 1
Code Generation Demo Configuration 1: Boards and Connection
Code Generation Demo Configuration 3: Two PCs Test
Demo for Architecture Template: 19 March 2004
Equipment Health Monitoring Hardware Diagram
Local IO System for Gas Turbine, Gearbox and Waterjet
Local IO System for Gas Turbine, Gearbox and Waterjet: Pressure Sensors
Local IO System for Gas Turbine, Gearbox and Waterjet: Speed Sensors
Local IO System for Gas Turbine, Gearbox and Waterjet: Temperature Sen
Marine Application with Five Propulsion Trains
Marine Application with One Propulsion Train
Marine Application with Three Propulsion Trains
Marine Demonstrator: Bluetooth CAN/Ethernet
Marine Demonstrator: Bridge Control Console
Marine Demonstrator: Five Trains Context
Marine Demonstrator: Five Trains Hardware
Marine Demonstrator: Local Controller Gas Turbine Gearbox and Waterjet

	Source	Results
Functionality of Control	465	High
Intelligent Transducer/Sensors	270	High
Percentage of Control Functionality that is SIL1	8	0-33
Percentage of Control Functionality that is SIL2	44	34-66
Percentage of Control Functionality that is SIL3	20	0-33
Percentage of Control Functionality that is SIL4	3	0-33

Hardware Name	Number of Used	Redundancy	Safety Level	Intelligent
Normal PC	12	1	1	FALSE
Normal PC2	2	1	1	FALSE
Power PC2				FALSE

Fig. 3: Example of Reliability and Through-Life-Cost Calculation

The Co-simulation environment uses middleware technology to integrate the COTS tools and the MAA tools. Originally, the middleware used was *Common Object Request Broker Architecture* (CORBA)[3], however in 2004, the new Object Management Group (OMG)'s middleware standard Data Delivery Service (DDS) based on a publisher-subscriber design became available and it was a mandated standard for use across the Department of Defence (DoD) in the USA. The first DDS-compliant product was *Network Data Delivery Service* (NDDS) by RTI [4]. Since then, NDDS has been adopted for marine and aerospace applications in the USA. Thus, NDDS was recently adopted within the FLEXICON MAA toolset instead of CORBA. Also, timing performance results carried out for the Marine application by the University of Sheffield showed that NDDS outperforms CORBA for both small and large data sizes.

Co-simulation is one of the best ways to develop complex systems as it allows parts of the system to be integrated and tested through virtual prototyping [5]. The Co-simulation environment targeted to MAA applications consists of already made Application Programme Interfaces (API) based on the NDDS middleware technology. Interfaces currently available within the toolset for the following COTS tools are: Simulink/Stateflow, ISaGRAF, ARTISAN RtS and Microsoft Excel. Using NDDS as a middleware over the Ethernet, it is possible to easily add a new COTS tool or hardware device. Fig. 4 shows the vision of the Co-simulation environment with the integration of *N* COTS tools, the MAA tools and *N* hardware devices in the loop.

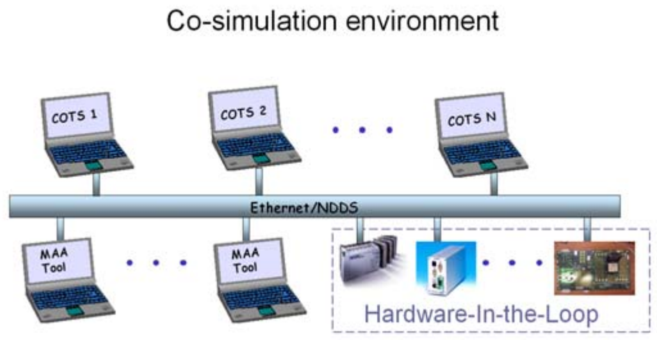


Fig. 4: Co-simulation environment

2.2 Code Generation

In the Implementation Phase, the hardware and software designs are implemented. This includes the code generation. Fig. 5 shows hardware and software implementation in UML (ARTISAN RtS tool), the dynamic models implemented in Simulink and ISaGRAF. After this implementation, code generators are used to finally produce target specific code. The MAA code generators used are the Java Code Generator and the VHDL Code Generator, which can generate code either from Simulink or ARTISAN with traceability and consistency checking previous to code generation. In order to generate C code, e.g. a controller in Simulink, the Real-Time Workshop is used together with the Tornado tool (VxWorks Real-time Operating System and NDDS communication). For the ISaGRAF tool, Target Independent Code (TIC) was automatically generated.

The Configuration Management tool (shown in Fig. 5) allows saving or retrieval of information to and from the User Design Database. This information includes the requirements documentation, systems models, the reusable template libraries used by the MAA code generators, the code generated by the different tools and also different co-simulation configurations which are accessed by using the System Builder. The System Builder gives the flexibility to support different COTS hardware platforms.

Configuration Management

The Configuration Management Database tool is designed to provide the following information with regards to the management services in the FLEXICON MAA toolset. The Configuration Management can be accessed with Internet Explorer. The MAA Configuration Management Database implements secure control on the login service and each user requires a user name and a password in order to log into the database.

The MAA Configuration Management Database maintains the *Hardware-Specific Code Repository* that refers to all the code generated by the FLEXICON MAA code generation and COTS tools. Also, it contains code test information separated in two stages (*Unit Test* and *Integration Test*). A record of testing is highly important, especially when developing safety-critical code. This information includes documentation such as test plans, test results and associated test harnesses recorded. *COTS Tools Management* is also provided, since it is important to store information

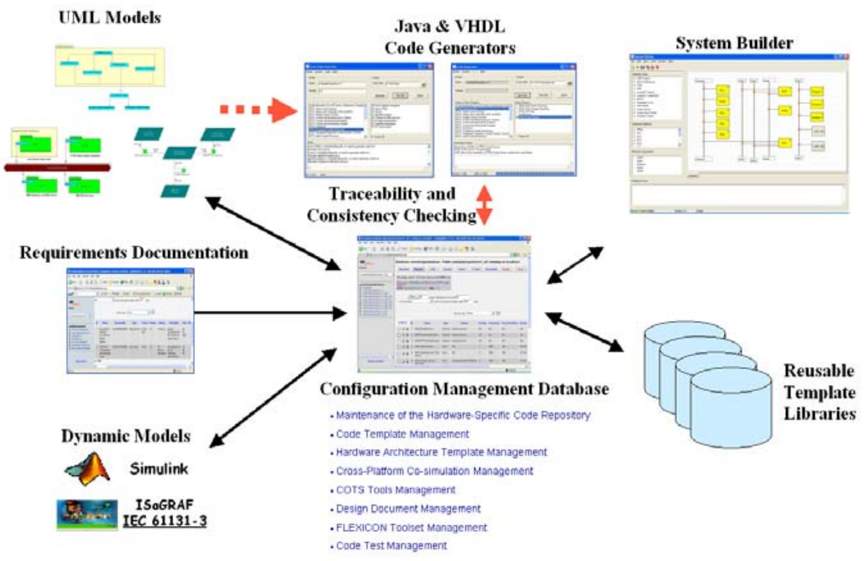

UML Models
Java & VHDL Code Generators
System Builder
Traceability and Consistency Checking
Requirements Documentation
Configuration Management Database
Maintenance of the Hardware-Specific Code Repository
Code Template Management
Hardware Architecture Template Management
Cross-Platform Co-simulation Management
COTS Tools Management
Design Document Management
FLEXICON Toolset Management
Code Test Management
Reusable Template Libraries
Dynamic Models
Simulink
ISaGRAF
IEC 61131-3

Fig. 5: Code Generation Phase

on the tools, which are currently available within the Co-simulation environment. It includes information such as the name and version of each software package, the operation systems on which they run, the supplier and a short description of the tool etc.

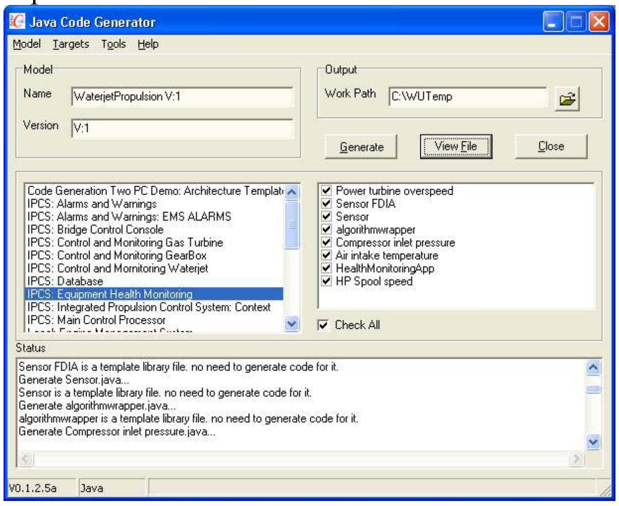
As part of the Configuration Management, the *Design Document Management* is also considered. This Design Document Management records the documentation associated with a specific co-simulation. Document ID, type, format, version number and author is recorded. A brief description of the documents contents is also given along with the date of issue. This should include information on the initial requirements and any information required to enable a specific co-simulation to be performed.

FLEXICON Toolset Management is also included within the Configuration Management. This contains the name and version of each MAA tool. The inputs/outputs of the tools and the development tools used to design each tool are recorded. This is particularly important if new versions of the tools are issued. Also, *Cross-Platform Co-simulation Management* is used to store information on different co-simulation models which have been developed. Within the database information of the different models issued within a particular co-simulation demonstration is recorded. Also, inputs and outputs from the models are stored along with the development tools and the model versions used. The target hardware platform and operating system is also recorded along with the destination of the source code.

Code Template Management contains information on the code templates which are used by the MAA code generation tools checking the availability of the code templates before generating code. Finally, Hardware Architecture Template Management is also provided. This allows users to maintain the hardware architecture templates used when generating hardware-specific code for a given board.

Java Code Generation.

The Java code generator was designed to generate Java code for a given design model. The design model can come from two sources: the SIMULINK model and an ARTiSAN class diagram. If starting from a UML class diagram the user can start the UML-to-Java code generation by choosing “ARTiSAN RtS” from the “Model” menu. If the user selects “Simulink”, the SIMULINK-to-Java code generation will be started instead. Fig. 6 shows the GUI of the FLEXICON MAA Java Code Generator for an ARTISAN class diagram. Previous to code generation consistency and traceability checking against the user requirements is performed.


Java Code Generator

Model Targets Tools Help

Model
Name: WaterjetPropulsion V:1
Version: V:1

Output
Work Path: C:\WUTemp

Generate | View File | Close

Code Generation Two PC Demo: Architecture Templat
IPCS: Alarms and Warnings
IPCS: Alarms and Warnings: EMS ALARMS
IPCS: Bridge Control Console
IPCS: Control and Monitoring Gas Turbine
IPCS: Control and Monitoring GearBox
IPCS: Control and Monitoring Waterjet
IPCS: Database
IPCS: Equipment Health Monitoring
IPCS: Integrated Propulsion Control System: Context
IPCS: Main Control Processor

☑ Power turbine overspeed
☑ Sensor FDIA
☑ Sensor
☑ algorithmwrapper
☑ Compressor inlet pressure
☑ Air intake temperature
☑ HealthMonitoringApp
☑ HP Spool speed

☑ Check All

Status
```
Sensor FDIA is a template library file. no need to generate code for it.
Generate Sensor.java...
Sensor is a template library file. no need to generate code for it.
Generate algorithmwrapper.java...
algorithmwrapper is a template library file. no need to generate code for it.
Generate Compressor inlet pressure.java...
```

V0.1.2.5a | Java

Fig. 6: Java Code Generator example

Example of Java Code Generation for Health Monitoring Web Service

A Health Monitoring (HM) Web Service was built as part of the system requirements of the Marine application. However, this service can be easily adapted for another application. The HM is used to control and monitor the application equipment using the Ethernet. The HM structure has been modelled in the ARTISAN tool using a class diagram. Thus, sensors can be added easily, and by means of FDIA and control algorithm libraries the functionality of the sensors is built automatically in Java using the MAA Java code generator. Once the health monitoring application is generated in Java, the compiled Java classes can be invoked by the Health Monitoring Web Service. In order to access the Web Service, a Web browser is used as shown in Fig. 7. This figure shows the Health Monitoring information for the Marine application. In addition to sensor data other parameters and trends can be displayed. This allows equipment to be operated optimally providing fuel economy, high reliability and availability.

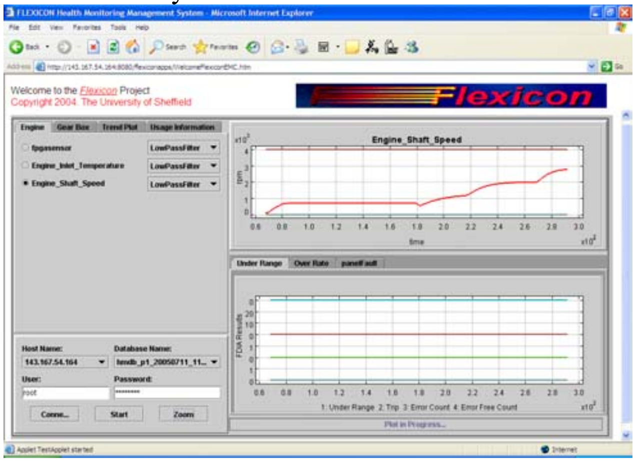
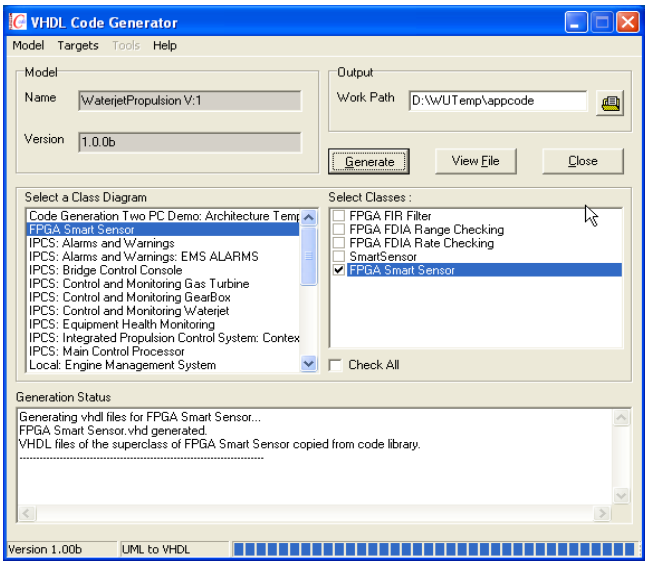


Fig. 7: Health Monitoring Using a Web Browser

VHDL Code Generation
The VHDL code generator tool is able to generate VHDL code for a specified XILINX FPGA device. Like the Java code generator there are two design sources that can be used for code generation: a SIMULINK model and an ARTiSAN class diagram. Simulink models can be designed using special XILINX FPGA blocks which are entered into Simulink. The code generator has integrated the XILINX System Generator in order to generate VHDL code from such a SIMULINK model. Once a model is selected, the VHDL Code Generator will invoke the XILINX System Generator to complete the rest of the process. In the case of ARTISAN, the class diagrams are used as the source for VHDL generation. Fig. 8 shows an example of the FLEXICON VHDL Code Generator. Consistency checking is also performed before code generation. If the consistency checking detects a problem the code generation process will be terminated immediately.


VHDL Code Generator
Model Targets Tools Help

Model
Name: WaterjetPropulsion V:1
Version: 1.0.0b

Output
Work Path: D:\WUTemp\appcode

Generate | View File | Close

Select a Class Diagram:
- Code Generation Two PC Demo: Architecture Temp
- FPGA Smart Sensor
- IPCS: Alarms and Warnings
- IPCS: Alarms and Warnings: EMS ALARMS
- IPCS: Bridge Control Console
- IPCS: Control and Monitoring Gas Turbine
- IPCS: Control and Monitoring GearBox
- IPCS: Control and Monitoring Waterjet
- IPCS: Equipment Health Monitoring
- IPCS: Integrated Propulsion Control System: Contex
- IPCS: Main Control Processor
- Local: Engine Management System

Select Classes :
- [ ] FPGA FIR Filter
- [ ] FPGA FDIA Range Checking
- [ ] FPGA FDIA Rate Checking
- [ ] SmartSensor
- [✓] FPGA Smart Sensor

[ ] Check All

Generation Status
```
Generating vhdl files for FPGA Smart Sensor...
FPGA Smart Sensor.vhd generated.
VHDL files of the superclass of FPGA Smart Sensor copied from code library.
-----------------------------------------------------------------------
```
Version 1.00b | UML to VHDL

Fig. 8: VHDL Code Generator example

2.3 Co-Simulation with Hardware-In-the-Loop

System Builder
The System Builder is a user-friendly software tool that enables design engineers to dynamically configure the co-simulation of a distributed control system. It also gives designers the capability to securely manage the various hardware platforms and software packages for system co-simulations. The System Builder allows adding, removing and modifying hardware platforms and software packages for a co-simulation project. This tool provides a “Software task manager”, a “Hardware Platform Manager” and a “Network Component Manager” to manage the hardware and software packages. The hardware and software packages are managed on each project basis; however, they can be reused by a new project by simply adding them into it.

In order to build a distributed system architecture, the System Builder provides a graphical tool that helps users to build the system architecture for a distributed system. The user simply drags and drops the various hardware and network components into a drawing board. The System Builder will detect the connections automatically and update the design information accordingly. Using the System Builder it is possible to assign dynamically software tasks to a selected hardware platform. Thus, multiple software tasks can be allocated to a hardware platform by simply dragging and dropping them into the box in the drawing board.

Another functionality of this tool is the uploading of software tasks over the distributed network. This means that software tasks of a co-simulation project can be uploaded to the corresponding hardware platforms automatically through the distributed network. This is very useful when the distributed network covers a large area and there is long distance between the configuration machine and each hardware platform. For instance, the designer does not have to go to each node and copy files into it. This functionality is particularly useful when security is considered in the system configuration, since all software tasks are uploaded automatically and finger errors of the designer can be avoided. Fig. 9 shows a distributed architecture built using hardware and network components available in the System Builder.

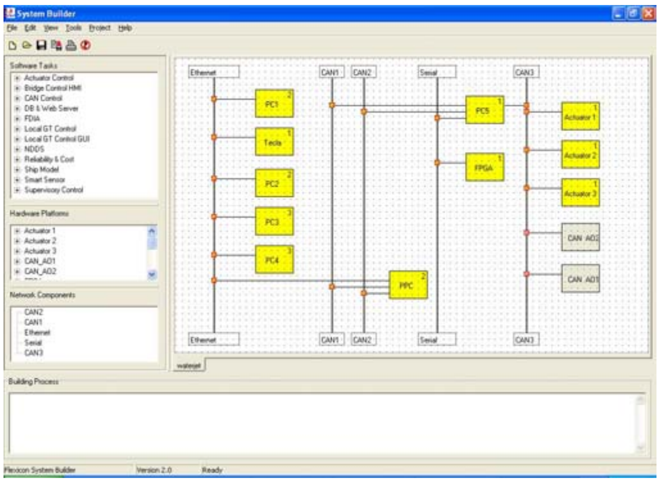


Fig. 9: Example of a Distributed System Architecture Using the System Builder Tool

Other support for Hardware-In-the-Loop Co-Simulation
In order to support Co-simulation with Hardware-In-the-Loop, a CAN Controller tool and a Wireless Extractor tool are provided. The *CAN Controller*

monitoring tool provides the current status on a duplex CAN bus network. This tool allows checking for any failure in the CAN network. CAN messages for each network are displayed. In addition, the waterjet actuator position data is displayed. This allows tuning the boat parameters (steerable motor speed, main motor speed and steering angle). The tool connects to the CAN bus via a USB-to-CAN interface. Also, the tool allows the transmission of CAN data to a Serial port to be processed by another processor such as a FPGA and return it back to the CAN network. *Wireless Data Extractor* tool reads wireless sensor data that have been recorded online and saved into the application database. Thus, wireless data is available from the Health Monitoring Web Service.

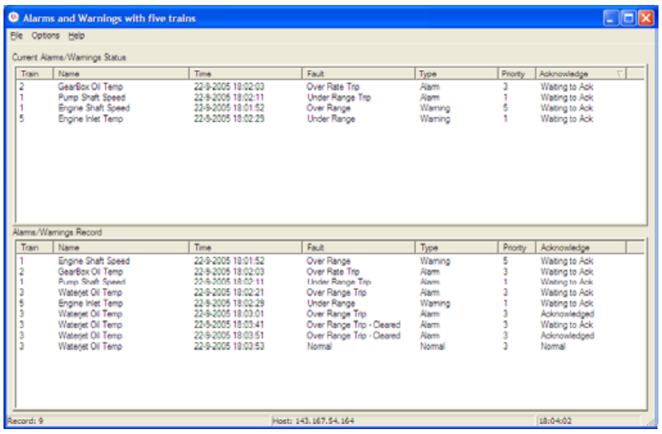
2.4 Human Machine Interfaces

To drive the co-simulation in a realistic manner for a particular application. It is necessary to support the user with a HMI that is familiar. In the case of the Marine application, HMIs were developed for the Bridge Control Console, Switchboard Control Console and Local Control. All of these displays are very application specific, however one interface was developed that has a wider application; this is the *Alarms and Warnings Management* tool.

Alarms and Warnings Management

To record the alarms and warnings that occur in a distributed system, an alarms and warnings management tool was developed. When alarms and warnings take place in the system, these are displayed automatically in this management tool and logged automatically into the application database.

Fig. 10 shows an example of the alarms and warning management tool for the Marine application where the propulsion train number, name of the sensor, date/time stamp, fault type, priority of the alarm/warning and the status of acknowledgment were recorded during the testing phase.



Current Alarms/Warnings Status

Train	Name	Time	Fault	Type	Priority	Acknowledge
2	GearBox Oil Temp	22-9-2005 18:02:03	Over Rate Trip	Alarm	3	Waiting to Ack
1	Pump Shaft Speed	22-9-2005 18:02:11	Under Range Trip	Alarm	1	Waiting to Ack
1	Engine Shaft Speed	22-9-2005 18:01:52	Over Range	Warning	5	Waiting to Ack
5	Engine Inlet Temp	22-9-2005 18:02:29	Under Range	Warning	1	Waiting to Ack

Alarms/Warnings Record

Train	Name	Time	Fault	Type	Priority	Acknowledge
1	Engine Shaft Speed	22-9-2005 18:01:52	Over Range	Warning	5	Waiting to Ack
2	GearBox Oil Temp	22-9-2005 18:02:03	Over Rate Trip	Alarm	3	Waiting to Ack
1	Pump Shaft Speed	22-9-2005 18:02:11	Under Range Trip	Alarm	1	Waiting to Ack
3	Waterjet Oil Temp	22-9-2005 18:02:21	Over Range Trip	Alarm	3	Waiting to Ack
5	Engine Inlet Temp	22-9-2005 18:02:29	Under Range	Warning	1	Waiting to Ack
3	Waterjet Oil Temp	22-9-2005 18:03:01	Over Range Trip	Alarm	3	Acknowledged
3	Waterjet Oil Temp	22-9-2005 18:03:41	Over Range Trip - Cleared	Alarm	3	Waiting to Ack
3	Waterjet Oil Temp	22-9-2005 18:03:51	Over Range Trip - Cleared	Alarm	3	Acknowledged
3	Waterjet Oil Temp	22-9-2005 18:03:53	Normal	Normal	3	Normal

Record: 9   Host: 143.167.54.164   18:04:02

Fig. 10: Alarms and Warnings Management Tool example

3 Marine Application Portable Demonstrator

To provide a challenging and complex application a high-speed vessel propulsion system has been considered. Rolls-Royce Marine provided the University of Sheffield with the functional and non-functional requirements of their new generation of jet-powered high-speed vessels. The example vessel utilizes five propulsion trains and a sharp long hull that allows the ship to cut through waves at 50 knots rather than ride over them (see Fig. 11). This allows the ship to maintain higher speeds than conventional ships even in mid-ocean and bad weather. In such a vessel crossing the Atlantic should take around 90 hrs whereas existing “fast” vessels (32 knots) take around a week [2]. The vessel consists of five propulsion trains, each one with a MT30 gas turbine, a Kamewa waterjet and a gearbox. This type of vessel is controlled by a complex, distributed, real-time, safety-critical Integrated Propulsion Control System (IPCS) for control and monitoring.

The IPCS allows the operator to control and monitor all the propulsion train equipment as an entire system from the two control consoles (Bridge and Switchboard). This supervisory control system at the top level interacts with local equipment controllers that control the gas turbines, gearboxes and waterjets. The equipment is controlled by a Local Control console which can control starting, normal stop, shutdown and speed of the gas turbines, speed of the waterjets and their direction, as well as engagement and disengagement of the gearboxes.

Fig. 11: High-speed vessel

3.1 Hardware Description

For the Marine Application the portable demonstrator can be configured as shown in Fig. 12. The demonstrator consists of several Pentium 4s (PCs and laptops) used to execute some of the COTS tools (Simulink/Stateflow and ARTISAN RtS), two HMIs (Bridge/Switchboard Control Console and Local Control – PCs with Touch Screens), the MAA tools (Reliability & Through-Life-Cost, Health Monitoring Web Service and

CAN Controller-comparator/monitor) and a virtual simulation of the ship. Another processor used is the Tecla PLC that contains the Supervisory Control program that communicates with the HMIs. The Supervisory Control software was implemented in the ISaGRAF tool and downloaded directly into the PLC. The control algorithm of one propulsion train runs in a PowerPC processor under the VxWorks Real-time operating system. This control was modelled first in the Simulink tool and then, C code was generated using the RTW. The C code also includes the VxWorks and NDDS functionality. A smart sensor runs in a FPGA board. This board communicates to the PowerPC via the CAN Controller tool which transforms Serial data into CAN data. The CAN Controller also takes CAN messages from a duplex CAN network and compares these messages for fault tolerance purposes. The validated CAN messages are then sent to the scaled model boat via a remote control. Wireless sensors are also included in the demonstrator, these are located in the boat. The COTS tools and the MAA toolset communicate with each other via the Ethernet using the NDDS middleware. Finally, a joystick is used by the captain to input the steering, reverse and throttle commands to the virtual vessel.

3.2 Software Description

- ISaGRAF Enhanced is used in the Supervisory Control program
- Simulink is used for continuous model simulation of the ship dynamics, simulation of four propulsion trains and fault injection
- Stateflow for fault detection, isolation and accommodation
- ARTISAN for the hardware architecture modelling and reliability calculations, also for model functionality definition
- Microsoft Excel has been used for Reliability & Through-Life-Cost (R&TLC) calculations
- HMI of ISaGRAF for the Bridge Control Console
- C++ .Net for the Local Control Console
- Simulink, RTW, VxWorks and NDDS communication for the controller of one propulsion train
- Virtual ship navigation simulation
- Web Service technology for the Health Monitoring
- ICHMMonitor for the Bluetooth Wireless Oceana sensors

To operate the vessel then, the operator at the bridge has a touch screen (Bridge Control Console) and a joystick. Using these interfaces, the operator can operate the vessel in different modes, control the speed of the propulsion trains, steering and reverse bucket of the waterjets, and pass control to the Local Control operator. The operator at the Local Control also has a touch screen that allows him/her to control each of the propulsion trains separately. The Local Control controls the gearboxes, gas turbines and shaft state. Also, the local operator has control over the maintenance mode and can request control from the operator at the Bridge. The joystick interface connected to the Virtual Navigator tool helps the operator at the Bridge to navigate the vessel. The dynamic

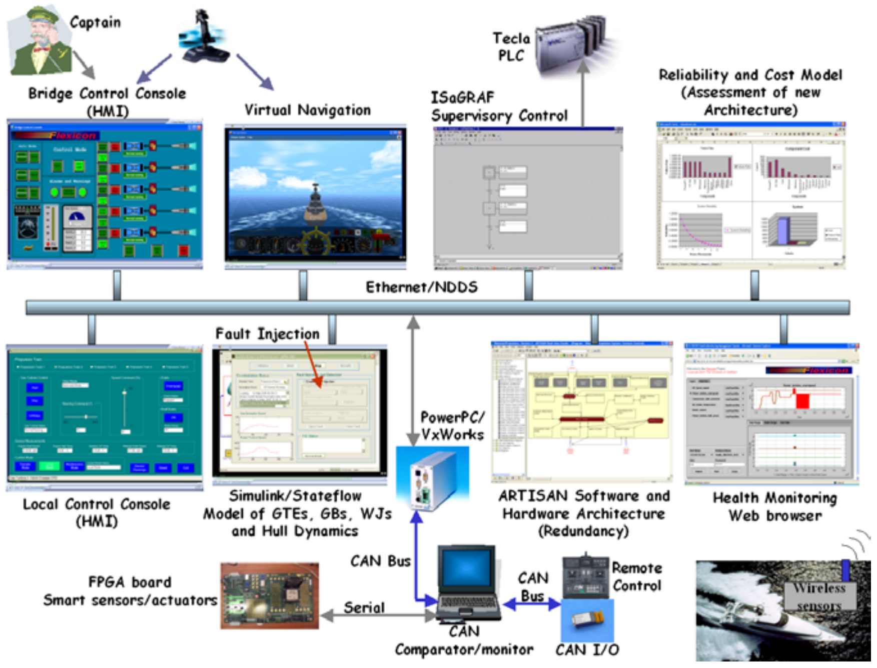

Captain
Bridge Control Console (HMI)
Virtual Navigation
Tecla PLC
ISaGRAF Supervisory Control
Reliability and Cost Model (Assessment of new Architecture)
Ethernet/NDDS
Fault Injection
PowerPC/ VxWorks
Local Control Console (HMI)
Simulink/Stateflow Model of GTEs, GBs, WJs and Hull Dynamics
ARTISAN Software and Hardware Architecture (Redundancy)
Health Monitoring Web browser
CAN Bus
FPGA board Smart sensors/actuators
Serial
CAN Comparator/monitor
CAN Bus
Remote Control
CAN I/O
Wireless sensors

Fig. 12: Marine Portable Demonstrator

simulation of the ship is controlled by Simulink. Fault injection and the simulation of propulsion trains are also executed in Simulink. The boat's waterjets are controlled via the code running in the PowerPC. For this, CAN messages are transmitted to the remote control of the boat. A smart sensor (FPGA) filters the raw data and sends it back to the Health Monitoring Web Service. Apart from the sensor data, other parameters and trends of the propulsion system are displayed in the Health Monitoring that will help the maintenance operator to monitor the equipment. The architecture model of the application is displayed in the ARTISAN tool and this information is used for the R&TLC calculations.

Results collected during the demonstrator activities showed that a reduction of 30% in the developing time can be achieved using the MAA toolset. The use of an automatic code generation has a significant impact on the coding phase but also co-simulation improves requirements production and with the Hardware-In-the-Loop testing time is greatly reduced.

4 Concluding Remarks

In this paper, the development process used to produce a Marine application utilising the FLEXICON MAA toolset has been presented. In order to achieve a successful exploitation and to illustrate more realistically how the MAA toolset can be used to produce the virtual rapid prototyping of the marine application, a demonstrator and a portable demonstrator were built. During the demonstrators activities data were gathered showing that an important reduction of time can be achieved by implementing the initial requirements using the MAA co-simulation environment, then modelling with co-simulation, coding and testing again with co-simulation and Hardware-In-the-Loop.

Acknowledgements
The authors gratefully acknowledge the financial support of the European Union's Information and Science Technologies programme for the FLEXICON project IST-2001-37269 and Rolls-Royce plc. Additionally, the authors gratefully acknowledge the support provided by ARTISAN, Borland and WindRiver.

References:

[1] Thompson, H.A., D.N. Ramos-Hernandez, J. Fu, L. Jiang, I. Choi, K. Cartledge, J Fortune and A. Brown. (2006) A Flexible Environment for Rapid Prototyping and Analysis of Distributed Real-Time Distributed Safety-Critical Systems. Accepted for the Control Engineering Practice Journal.

[2] Thompson H.A., D.N. Ramos-Hernandez, K. Cartledge, J. Fortune and A. Brown (2003). A Flexible Control Systems Development and Integration Environment for Distributed Ship Control, 13th International Ship Control Systems Symposium (SCCS), Orlando, USA, April 7-9.

[3] http://www.omg.org/gettingstarted/corbafaq.htm Access date: 06/06/05.

[4] http://www.rti.com/index.html

[5] Thompson, H.A. and D.N. Ramos-Hernandez (2004). A Flexible Control Systems Development and Integration Environment for Distributed Control. Proceedings Sixth Portuguese Conference on Automatic Control, Controlo 2004, Faro, Portugal, June 7-9.

[6] http://www.mathworks.com/products/simulink/

[7] http://www.isagraf.com/

[8] http://www.artisansw.com/