Model Driven Development of a Distributed Marine Control Application Using the Unified Modelling Language

L. Jiang, H.A. Thompson, D.N. Ramos-Hernandez, and J. Fu

Rolls-Royce University Technology Centre in Control and Systems Engineering, Department of Automatic Control and Systems Engineering, University of Sheffield, UK

Keywords: Unified modelling language, model driven development, distributed control systems, marine application

Abstract

Developing and modelling an industrial distributed control system is a challenging task, as a distributed solution will increase the level of complexity. An Object-Oriented (OO) methodology has proven to be a powerful technique with respect to modelling, analysis, and design of complex systems in the software engineering field. The goal of this paper is to explore the application of a model-driven development methodology for a distributed control system in an object-oriented environment using the Unified Modelling Language (UML). It is demonstrated that UML can be used to model a marine application without being associated with other modeling tools, covering both hardware and software domain implementations, and formulating a platform for developing reusable, extendible, and modifiable software for a distributed control system.

1 Introduction

Currently industry is interested in developing a distributed control systems due to the increasing performance of embedded processing power and the decreasing of hardware cost [2,7]. This offers many potential advantages through reduction in wiring and ability to build in additional processing power such as fault diagnostics into the distributed elements. However, a distributed solution will cause an increased level of complexity, which will cause a significant technical risk and a higher development cost when designing a fault tolerant system. On the other hand, to make the system affordable, there is increasing emphasis on utilisation of Commercial Off-The-Shelf (COTS) technologies. This will also cause the system integrator to be presented with the need to support a number of different equipment from different manufactures.

In the software engineering field, an OO methodology has proven to be a powerful technique with respect to modelling, analysis, and design of complex system. UML has emerged as a standard notation for the conceptual and visual modelling of OO software [5]. It can offer reusability, extendibility, and modifiability in software design. Due to the increasing complexity and flexibility requirements, the OO technology has also been employed to develop software for industry distributed control system, such as process control [4], and flexible manufacture control system [3,7]. However, there is still an open problem to develop an open and flexible platform to integrate currently available COTS equipments for a distributed control system.

The European Union funded project ‘Flexible Control Systems Development and Integration Environment for Control Systems (FLEXICON) project IST-2001-37269 [6] intends to investigate a toolset, which can help integrate equipment from different manufacturers, and investigate the overall performance of distributed systems. The FLEXICON project plan is developing the toolset based on an OO methodology and has chosen UML to model the application and provide a common integration interface. This will result in a model-driven development process for the distributed control system, in which all individual components are developed and integrated together based on the application’s UML model. The inclusion of modelling of hardware components in the whole system UML model is implemented by using a CASE tool called as ARTiSAN Real-Time Studio (RtS) [1].

This paper illustrates the UML modelling procedure of a distributed waterjet propulsion system for a high-speed merchant vessel. At first, a development procedure is proposed and then the application is modelled via different UML diagrams for describing physical distribution of hardware, software implementation and mapping between software and hardware.

2 UML

UML is a graphical design and modeling language that specifies semantics and notation, which embraces all the features of the OO paradigm such as reusability, representational versatility, inheritance property, and rapid prototyping. Diagrams provided by UML can be grouped as use-case view diagrams, dynamic view diagrams, and object view diagrams, and can be used to develop three orthogonal models: usage, object, and dynamic models. The usage model captures interaction between actors and the system and what the system does based on outside requirements. The object model represents the static, structural aspects of the system, in which objects, their attributes, their operations, and their relationships with other objects are represented in detail. The dynamic model represents the temporal behavioural aspects
of the system, in which any change is described by activities and events, sequences of events, and states between objects or inside a complex object.

ARTiSAN RTs is an UML tool and it uses the industry-standard UML notation to model many aspects of a system’s intended functionality and software object architecture. Built on top of this foundation, it provides additional tools for modelling the aspects of a system that are unique to real-time systems, such as the tasking model, the system's hardware architecture, the mapping of objects to tasks, etc [1]. The procedure of building a model within UML is summarized in Section 4.

3 The Marine Application

The FLEXICON marine application has been designed based on an Integrated Propulsion Control System (IPCS) for high-speed merchant vessels. This application provides diverse requirements for the application of distributed process control and real-time safety-critical control, featuring complex real-time constraints and strict performance targets [2].

The propulsion system for the ship utilises five MT30 gas turbines, five Kamewa waterjets, and associated gearboxes including clutches and fluid couplings. The gas turbine and waterjets are arranged in five propulsion trains with a single gas turbine driving a single waterjet through a reduction gearbox and clutch. The main functions of the IPCS include: control and monitoring of the gas turbines; control and monitoring of the waterjets; control and monitoring of the gearboxes; management and monitoring of alarms, warnings and status information; and equipment health monitoring.

The IPCS will implement the main functionality to allow the operator to control and monitor the whole propulsion train equipment by interfacing to the operator console. The IPCS provides both graphical user interface and operator input console devices. An Equipment Health Monitoring (EHM) system is also included. The whole configuration of the IPCS is shown in Figure 1 [2].

Based on the requirements of the marine application, a demonstrator will be constructed to evaluate the FLEXICON toolset. The main configuration of the demonstrator is shown in Figure 2. This demonstrator consists of two main PCs that connect via a duplex Ethernet network. The duplex Ethernet network is connected to local triplex CAN buses by using an Ethernet/CAN Gateway. A triplex CAN bus is used to connect the local control system for the waterjet propulsion system (including the gas turbine, gearbox and waterjet). The two PCs are designed to run the main control functionality of IPCS, which correspond to the ‘Default Processor Unit’ and ‘Backup Processor Unit’ in the real system. At the local-level, the local gas turbine controller (Engine Management System (EMS)) will be implemented based on a VxWorks running on PowerPC, while PLCs are planned to control the gearboxes. Local microcontrollers (C167) are used to control the waterjet actuators providing a set of smart actuators.

A wireless equipment monitoring system will be also implemented based on Bluetooth and Wireless Ethernet protocols.

4 Design Process

As UML is a modeling language that only specifies semantics and notation, no process methodology is designed within it. Thus, a design procedure for the marine application has been proposed, as shown in Figure 3. The whole UML modeling process starts with the Use Case diagrams which are used to capture the main functionality of the marine application. System Architecture Diagram, which models the system’s physical structure and describes the hardware structure for the marine demonstrator. Class diagrams were created to depict the static behavior and the software structure of the application. The dynamic behavior of the application was modeled with corresponding State Diagrams, Collaboration Diagrams, and Sequence Diagrams. A Concurrency Diagram was used to describe the implementation with the real-time environment and system constraints were modelled to define the timing performance of the application. Due to page limits, this paper only reports a few parts of the marine application model, which mainly focus on the operation of gas turbine engine. The state diagram and real-time modelling are not reported.
4.1 System Architecture Diagram

The system architecture model which is specific to the ARTiSAN RTS tool describes the hardware information and the distribution of the application. The architecture diagram uses the following components: subsystem, interface device, board, disk, multi-drop bus, and types for defined special characteristics for all the above components [1].

When the application code is produced, the system architecture diagram can be used to map software code to different hardware board. This makes it possible to produce a model that allows the functionality to be easily partitioned onto different hardware architectures. When providing systems to different customers the actual hardware choice (PC-based technology, PLC-based technology, PowerPC based technology, data bus standards, etc.) is often driven by customer preferences. Thus, it is important to model the functionality at a level that allows it to be moved between different hardware platforms.

The top-level architecture diagram of the marine demonstrator is shown in Figure 4. This diagram describes one simplified solution for the demonstrator based on Figure 1 and Figure 2. It consists of the following boards: a Bridge Control Console (PC board), a processor unit (TECLA PLC board), a Gateway (Ethernet/CAN board), a Health Monitoring (PC board), a PowerPC board for local controllers of the gas turbine, gearbox and waterjet, and another PowerPC board for the real-time simulator of the IPCS system, respectively. The ‘TECLA PLC’ and ‘Bridge Control Console PC’ are connected via a duplex Ethernet network to the Gateway (Ethernet/CAN). The local controllers and their I/O systems communicate via a triplex CAN network. The ‘Health Monitoring PC’ connects with the local I/O subsystems and the Real-time Simulator via a separate industrial Ethernet network. The local gas turbine control is implemented on an Embedded PowerPC (PPC405) board.

4.2 System Usage Modelling

System Usage Modelling is a useful tool for analysis, capturing, and specification of system functional requirements during the analysis phase of a project. There are two levels of Use Case: requirements analysis and system design. At requirements analysis a Use Case consists of a Use Case Diagram, plus a set of descriptions. The Use Case diagram identifies a set of Use Cases whose descriptions specify the sequence of interactions that take place between the actors and the entire system, as represented by the outer system boundary on the system architecture model. The design level of System Usage Modelling is treated as part of interaction modelling. A sequence diagram can be created for each Use Case, where the ‘objects’ are the actors, devices and subsystems contained within the system, and the interactions are the signal events that pass between these elements. These ‘system interaction diagrams’ are useful in helping exploration and refinement of their parent Use Cases, but are not an essential part of the Usage Model.

The IPCS operates the whole ship in three main modes: automatic operation mode, semi-automatic operation mode, and manual operation mode. It also implements the following functionalities: built-in test, emergency stop, fault accommodation and equipment health monitoring. These requirements are described in Figure 5. The Use Case ‘Automatic Operation Mode’ includes three sub Use Cases: Sprint mode, Cruise mode, and Heavy Sea mode, respectively.
At the local-level, the operation of the gas turbine engine can be described with four Use Cases: Start-up, Normal Operation, Stop and Emergency Stop, as shown in Figure 6.

**Figure 5: Use Case Diagram of functionality of IPCS**

**Figure 6: Use Case Diagram of control of gas turbine engine**

### 4.3 Object modelling via class diagram

Class modelling is the critical activity in object-oriented development. A class model sets the underlying foundation upon which objects will be put to work. Every use case described in the last section is investigated in order to find the object that takes part in it and the message exchanged between these objects, which will result in several collaboration diagrams. Once the different objects are extracted from these collaboration diagrams, the classes of the system can be proposed as a generalization of the objects. Finally, a class diagram will represent the static characteristic of the class and relationships among classes.

Based on the Use Case diagram shown in Figure 5, the ‘Main Control Processor’ package is defined, which includes a class “main control processor” as shown in Figure 7. It has seven aggregated classes, such as ‘auto-mode’, ‘semi auto-mode’, etc. These classes are connected with the ‘main control processor’ class via the “Aggregated Associations” relationship.

**Figure 7: Class Diagram of main control processor in IPCS**

Based on the Use Case diagram shown in Figure 6, the class diagram of the control of the gas turbine engine (Engine Management System, EMS) is shown in Figure 8. The main class ‘EngineManagementSystem’ is associated with several classes: ‘Communication Interface’ for communication, ‘LocalControlConsole’ and ‘Local Display Console’ classes for the local control console operation, and it also includes two classes describing the control law of the gas turbine: ‘Simple Gas Turbine Controller’ and ‘Complex Gas Turbine Controller’, respectively.

**Figure 8: Class Diagram of local gas turbine control system**

### 4.4 Dynamical modelling via sequence diagram and collaboration diagram

Apart from state diagrams, other diagrams that also depict system dynamics are the Sequence Diagram and the Collaboration Diagram. These diagrams both describe the flow of messages between objects. Sequence diagrams focus upon the timing order in which the messages are sent. The Collaboration diagram, on the other hand, focuses upon the relationships between the objects.

A Sequence Diagram shows the interactions, which occur between the objects in the Use Case. It is a graphical description of a Use Case. The diagram comprises statements, objects, and stimuli. A Collaboration Diagram models the structural dimension of a scenario by graphically
illustrating the interactions between actors and objects. It provides an alternative representation of a Sequence Diagram, each one modeling a different execution thread (or "scenario") through a Use Case by showing the flow or sequence of messages between associated objects. The startup process of a gas turbine is investigated and shown in Figure 9.

![Diagram](image)

**Figure 9: Collaboration Diagram of start-up gas turbine**

### 4.5 Mapping software to hardware

Concurrency Modelling is a vital part of real-time system design. It is based on the requirement to take a set of objects implementing the required functionality of a system and to schedule them so that the system meets its performance goals given the target hardware. It provides a set of techniques for analysing that the functionality defined by the objects in the system is executed simultaneously within the required performance constraints.

The Concurrency Diagram for the ‘Local Control of Gas Turbine’ is shown in Figure 10. This diagram contains four tasks. The main task is ‘Engine Management System’ which deals with the management and dispatching of the local control system, and calculating the corresponding control output based on measurements and operating command. The task ‘Communication Interface’ is responsible for sensor sampling and control output which are transferred via the CAN bus. There are also two tasks for local console input and display: ‘Local Console Input’ and ‘Local Console Display’.

![Diagram](image)

**Figure 10: Concurrency Diagram of control of gas turbine**

Each task can specify the linked objects and board, which will implement the mapping from software to hardware. Operations and events were specified between tasks and interface devices after the tasks had been linked to the corresponding objects.

### 5 Conclusions

This paper addresses the modeling process of a distribution control system for a marine application using UML of the ARTiSAN RtS tool. A design procedure is proposed for the whole development process. Following the proposed design procedure, and the requirement document, the marine application was modelled using different UML diagrams. This paper focuses on the building up an integration platform based on the UML model. As this is the early stage of the project, the development of the UML model of the application is of a top-level abstract model. Future detailed work is required for the final implementation.

### 6 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.

### 7 References