

Proof-Based Engineering of Real-Time Applications: An RTEdge™ Case Study

Roy Sarkar

Edgewater Computer Systems, Inc.

Copyright © 2009 SAE International

ABSTRACT

Edgewater's RTEdge™ Platform toolset is a model driven development environment for mission critical real-time systems. Using precise execution semantics and mathematical proof-based analysis, RTEdge™ enables the verification of critical properties of systems with high assurance.

This case study will follow the design and implementation life-cycle of a system representing a real-world, mission critical domain: airborne electronic warfare. Using examples and constraints taken from this system, software components will be built to illustrate the principles of architectural conformance, timeliness and testing as executed within a static analysis framework.

Using RTEdge™ as an example, this case study will introduce the concepts of model driven development in software and demonstrate how static analysis can be used to verify characteristics of a system that are traditionally left for later stages of development. A demonstration of interactive testing will be performed to illustrate how complex functional and performance testing can be achieved using information already present in the model.

INTRODUCTION

The Electronic Warfare System (EWS) consists of two sensors, a Missile Approach Warning System (MAWS) and a Radar Warning Receiver (RWR), a centralized manager (Electronic Warfare Controller) and a display device (Cockpit Display Unit) distributed over multiple computing platforms. This architecture is shown in Figure 1.

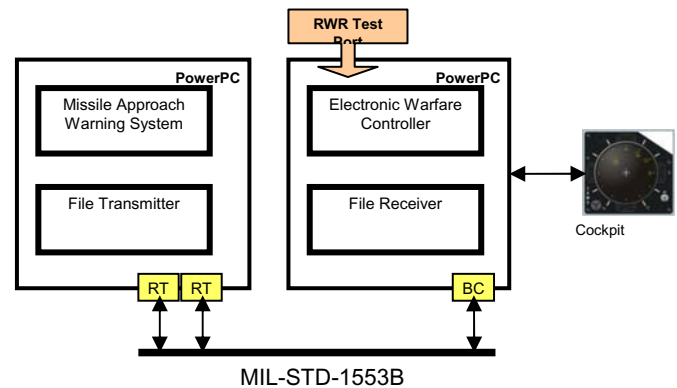


Figure 1: EWS Architecture

The purpose of the Electronic Warfare System is to detect entities in the environment that pose a threat to the host platform. This could be a potential threat, such as a radar emitter, or a definite threat such as an incoming missile. The sensors in the system pass entity data onto the Electronic Warfare Controller (EWC) which then determines whether the threat is valid and what severity classification it has. Once this is done, the EWC presents the threat to the user via the CDU. In parallel with this system is a data file transfer application representing lower priority work (such as mission data file transfer) competing for the same resources.

The software for the Electronic Warfare System was architected, designed, implemented and tested using the RTEdge™ Platform toolset.

ARCHITECTING THE SYSTEM

Specifying the Electronic Warfare System required the laying out of its component parts and relationships in addition to defining the real-time constraints that would prescribe a design. This specification was derived from

system requirements captured in the Telelogic DOORS[®] requirements management tool. The logical architecture was realized using RTEdge[™] Capsule Interface and Composite Capsule elements which described both the communications specification and software containment relationships. An example component is illustrated in Figure 2.

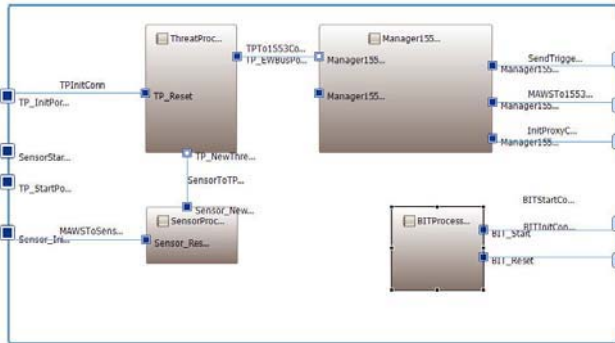


Figure 2: Component Specification and Architecture

Using a modeling language with well-defined constructs and properties enabled the architecture to be specified in a consistent and describable way. It also provided unique elements for which requirements could be traced to. The key RTEdge[™] modeling elements of Signals grouped into Protocols, mapped into Ports which are grouped into Capsule Interfaces are essential for expressing the logical architecture. The EWS was broken down into two separate applications, one for the sensors and one for the Electronic Warfare Controller (as this was a demonstration project, the CDU was built using Microsoft[™] Visual Studio and hosted on a Microsoft Windows[™]-based PC).

The communications mechanism for this platform is typically MIL-STD-1553B¹ so model elements were created to represent the command and data messages using Protocols in RTEdge[™]. Also created were the functional blocks responsible for the setup, running and fault management of the interface cards.

SPECIFYING TIME CONSTRAINTS

Once the logical architecture was complete, real-time constraints were placed on the inputs and outputs of the system as defined by the modeling elements (Protocols and Signals mapped to Ports on Capsule Interfaces). Critical time constraints for threat processing and display generation were specified using the Flow concept in the RTEdge[™] Platform. The Specified Flow element prescribes a causal relationship between two Port:Signal pairs within a Capsule Interface (namely that Port1:Signal1 must cause Port2:Signal2) and a limit on the maximum execution time allowed. This constraint on execution time is known as a deadline in the RTEdge[™] Platform. Key deadlines in the system included the time allowed to process data from a particular sensor and the time allowed to display a contact classified as a threat.

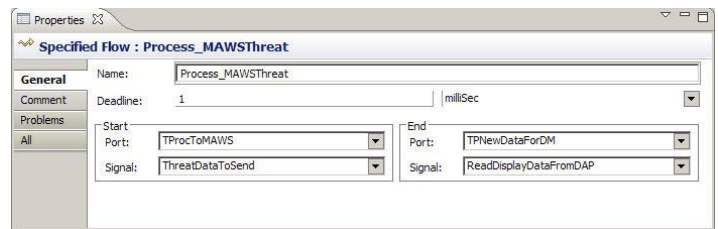


Figure 3: Properties of a Specified Flow

Once the architectural and time specification elements were created, they were linked back to the system requirements in DOORS[®] via the requirements linking tool supported by RTEdge[™] (EmbeddedPlus DoorKeeper for Eclipse).

After the specification model was completed and reviewed, it was necessary to break-up the model such that components could be assigned to different people for development. This breakdown was accomplished by assigning different Capsule Interfaces (the main specification element) to different developers and mandating that the design and implementation of each be carried out in separate models. This allowed each developer to work on their assigned capsules in isolation while maintaining the communications and time contracts specified by the Capsule Interface.

DESIGNING THE SYSTEM

With the architecture and specification complete, the model was passed to the developers for designing the individual components. With the RTEdge[™] Platform, the design elements are a refinement and extension of the specification elements within the same model. This results in a strict conformance between design and specification.

Further decomposition of the specified components was required and this was accomplished by realizing Capsule Interfaces with Composite Capsules. The Composite Capsule allows a specification of structure such that roles of other capsules can be contained. This provided a way to maintain a specification interface while decomposing the inner functionality. For this system, it was decided that a clear distinction be made between communications interfaces and the core sensor, threat processing and file transfer components. This would prove useful in both simplifying the system as well as insulating the different components from external changes.