Applying Model Driven Architecture to Radar Systems

Terri Potts[1], Stefanie Chiou[2], Gregory Eakman, Ph.D.[3]

(1)Integrated Defense Systems, Raytheon Company, Sudbury, MA 01776
E-mail: terri_potts@raytheon.com, Tel: (978)440-2387
(2) Integrated Defense Systems, Raytheon Company, Sudbury, MA 01776
E-mail: Stefanie_C_Chiou@raytheon.com, Tel: (978)440-2790
(3) PathFinder Solutions, Foxboro, MA 02035
E-mail: grege@pathfindermda.com, Tel: (508)265-5112

Introduction

Raytheon Company has been applying the use of Model Driven Architecture (MDA) to develop radar systems for the past three years. Pilot projects done on key radar systems components demonstrate MDA's applicability to real-time, embedded, and performance critical systems while providing other benefits such as increased productivity and product line components.

Background

MDA is a framework to specify applications and components as platform independent models (PIMs) and uses transformations to map these models onto target platforms. PIMs focus on the business logic and manage the complexity of systems through separation of concerns, not only between components, but also from the deployment platform, including programming language, operating system, communications, and distribution topology.

The PIM's higher level of abstraction and additional transformation step may also introduce inefficiencies unacceptable to high performance systems. However, the MDA framework and an open transformation environment allow tuned platform specific patterns and hand optimized code to be integrated to meet performance requirements.

This report covers three of the case studies applying MDA to radar components, a pilot project for an executive control component to evaluate the MDA process and tools, radar scheduling to assess strict performance requirements, and signal processing to address distributed parallel processing.

Fault Operability Processing

The initial pilot project on the executive control, health, and calibration components were a re-implementation of existing legacy code that were reintegrated with the system. The results of this project included a measured a significant increase in productivity and lower defect rates though integration and field tests.

Manual review of generated code concluded that while it may not be as good as the best real-time code writer could make it, it was better than the average team. The transformation rules for generating the code could also be extended to include new design and implementation patterns.

Radar Scheduling

MDA was then applied to the area of radar scheduling, a critical component of radar systems with strict performance and real-time requirements. This project's goal was to create a scheduler to be used in a product line, supporting a wide variety of scheduling requirements and running on multiple target platforms.

Scheduling is made up of three phases with different time horizons, long term planning, short term planning, and scheduling. Different radar programs in Raytheon Company's product lines have variations of scheduling requirements and algorithms. The scheduling PIM isolates the variations from the common processing using configuration parameters and overloaded operations. Markings and transformations optimize out the code and control paths not relevant to a specific target deployment.

Performance measurements were made on target platforms to determine the performance of the generated code. Measurements taken include total time (combining planning and scheduling), and individual times for scheduling. Multiple scenarios with different search and tracking parameters demonstrated sufficient performance on the target platforms. On two of the deployment platforms, the memory managers of the operating system were used by default, but were performance bottlenecks. Changes were made to the transformation rules to use an optimized memory manager, requiring no changes to the models.

As an example of the flexibility of the MDA, these models have been deployed on multiple hardware platforms and operating systems, in multiple languages, and with multiple deployment topologies. For example, in order to ensure that the planning did not interfere with the real-time deadlines, the scheduler was broken out from the planning algorithms into a separate, higher priority thread. This change also required no changes to the MDA model. Three key aspects of MDA enabled this flexibility – platform independence, objects, and state machines. Platform independence kept the model of the scheduling algorithms separate from the concerns of deployment topology. Objects representing planned schedule assignments are no longer accessed by the planning algorithms, enabling separation with a minimum of inter-task overhead. State machines express the algorithms in an asynchronous manner – a natural fit for event-driven, time-dependent systems. It is easier to implement an asynchronous design in a synchronous manner than it is to implement a synchronous design in an asynchronous manner.

Signal Processing

We then applied MDA to a signal processing project, one of the most complex parts of a radar system. The signal processor must convert raw data from the sensor into detection data for the tracker with real-time performance and latency requirements. In order to achieve the required performance, a radar signal processor must distribute its processing across multiple processors running in parallel.

The low level algorithms are written in C on top of already well optimized math libraries. Since this is where the bulk of the time is spent, hand tuning to the bit level to take advantage of CPU capabilities is well worth the investment.

However, signal processing algorithms comprise only a part of what the signal processor must do. A large part of the signal processor is dedicated to the management and movement of data between processors. We applied MDA to solve the data distribution problem, with an eye toward platform independence.

An MDA model of data flow model was built on platform specific mechanisms, providing a flexible framework to not only easily change target topology, but also the underlying transport mechanisms and operating systems. The separation of concerns allows highly tuned hand written code for the math algorithms to be integrated with generated code to achieve performance goals.

Conclusions

MDA has proved its worth in its application to these performance critical radar components. Platform independent models manage the complexity of developing radar systems and have shown large productivity improvements. Performance and real-time goals have been achieved through the use of open and customizable transformations tuned to the target platform and the ability to easily integrate hand optimized code.