An Early Look at the UML Profile for Schedulability, Performance, and Time for Engineering Large Scale Airborne C2ISR Platforms

Thomas M. Wheeler
The MITRE Corporation
twheeler@mitre.org

Abstract

This paper discusses an investigation into the utility of the Object Management Group's UML Profile for Schedulability, Performance, and Time (RT-UML Profile) for engineering a next generation of network centric, large-scale Command, Control, Intelligence, Surveillance, and Reconnaissance (C2ISR) aircraft. This activity applied the Profile in analysis of a multi-aircraft scenario, loosely based on DARPA's AMSTE research program. In this configuration, two aircraft, each with sensing capability, collaborate in real-time to develop precision targeting information for ground targets which is sent directly to in-flight munitions. This collaboration presents stressing, real-time requirements on a number of distributed, physically distinct platforms. The "RT-UML Profile" was used in conjunction with two commercial tools to perform analysis of the timing needs. Results of this activity indicate (1) the Profile assists engineers in developing better insight into temporal properties; (2) the Profile fits well within the DoD development processes; (3) use of sequence diagrams scales well for defining system load; and (4) the Profile should more directly supported threads that span software processes and physical resources.

Background

The goals for this activity were (1) become familiar with the basic technology involved with the UML Profile for Schedulability, Performance, and Time; (2) assess the maturity of vendor products for Profile support; and (3) assess the Profile's applicability for engineering large scale real-time distributed systems.

The U.S. Department of Defense (DoD) in recent years has emphasized the need to capitalize on the power of networking resources to improve operational effectiveness. Terms such as network centric warfare symbolize the importance placed by the DoD on achieving this objective. While better networking offers the strong potential for improved user benefit, it also presents a significant challenge to system engineers to take advantage of network resources while still delivering the high Quality of Service (QoS) the user has grown to expect and rely on with more closed solutions. A key reason this is a challenge is that when one buys into networking, one typically also accepts that resources will be shared by a larger community. Widely distributed resources and users with competing, real-time mission-critical objectives make engineering the next generation systems a daunting task. Given the difficulty in building previous, less networked systems to have satisfactory timing characteristics, it is clear we need better approaches if we are to have any hope of building the next generation ones.

The scenario examined in this activity was based loosely upon DARPA's AMSTE research program. Two aircraft, each with surveillance radars, work collaboratively to develop precision targeting information on ground-based tracks. This information is then sent to an in-flight munition and is used by the munition to strike a target.

Figure 1 shows a high level view of the system of systems. The aircraft in the middle has the responsibility of sensing, tracking, and updating the in-flight munition. The aircraft on the right is responsible for sensing and providing detections to the tracking function hosted aboard the first aircraft. The first aircraft delivers track information to the munition.

The basic steps in the process are for a sensor to "paint" a target, followed by signal processing to extract valid "detections", followed by a tracking function that updates estimates of a track's kinematics based on these detections, followed by the in-flight munition using this estimate to adjust its flight to strike a target. External communication via a radio is required between the two aircraft where one of them sends

detections to the other. External communication is also required between the aircraft performing the tracking function and the in-flight munition. Figure 2 shows this subsystem view including the information flows.

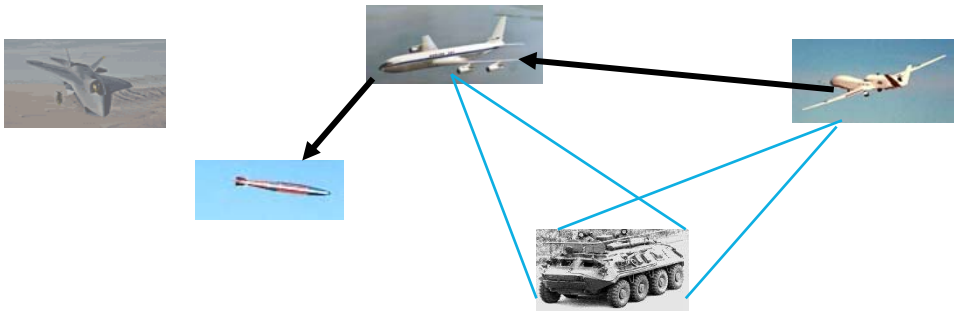

Figure 1. System Configuration

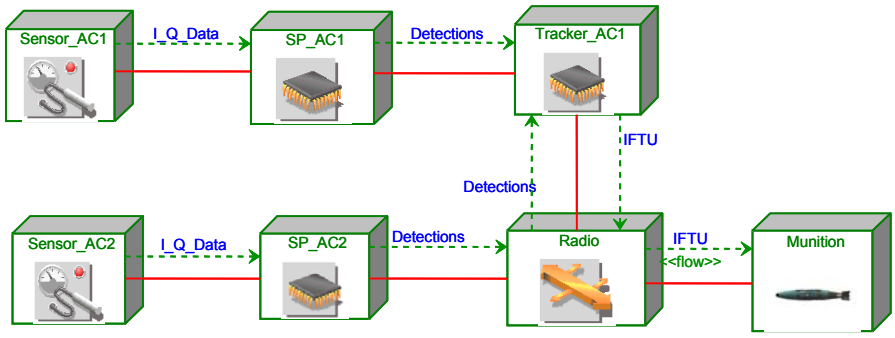

Figure 2. Subsystem Information Flows

Approach

The basic approach used was:

1. **Capture the design at a component or subsystem level using UML object and state diagrams.** The granularity chosen was driven by the level of knowledge we had of the internal details of the physical nodes. This model was made executable to verify the correctness of the captured design, but executability is outside the scope of the RT-UML profile and is not further discussed here.
2. **Define load scenarios using UML sequence diagrams.** One sequence diagram was used per logical system flow (which has similar characteristics to a Real-Time CORBA 1.2 (formerly 2.0) distributable thread). There were three of these distributable threads in the system. These threads spanned multiple UML active objects and physical resources such as processors. Each was modeled as being triggered by periodic stimuli. The first thread started in the sensor aboard the first aircraft and ran through the sensing, signal processing, and delivering of detections to the on-board tracker steps. The second thread started in the sensor of the second aircraft and was similar to the first with the added step of transmitting detections to the tracker via a radio link. Stimuli occurrence patterns for both of these threads were based on the characteristics of the sensors. The periods were not the same for each sensor thread. The third and last distributable thread began in the tracker and ran through the radio to the munition. Its period was based on the refresh rate needed by the munition. It should be noted that all three of the distributable threads shared, that is contended for use of, the tracker active object and two of the threads shared the radio active object.
3. **Add timing and resource information.** Information such as stimuli occurrence patterns, deadlines and execution times were added to the sequence diagrams using the RT-UML Profile. Software

objects were also mapped to hardware resources. Deadlines and occurrence patterns were based on information provided by a domain expert. Execution times were not known, but various values were used during the analysis step (see step 4 below) to understand the effect on the system's ability to meet its timing requirements.

4. **Analyze the results.** Rate and deadline monotonic analysis techniques were used, partially due to organic support for these techniques in the analysis tool used. Most of the analysis was done with a deadline monotonic algorithm that incorporated logic to evaluate a distributed system. Each of the distributable threads described above were defined with a single deadline and occurrence pattern, i.e., the deadline spanned multiple active objects which resided on separate processors. Execution times were at a finer level of granularity – specifically at an object's method level.

The tools used in this experiment were Rhapsody™ from I-Logix for the UML work and RapidRMA™ from Tri-Pacific for the analysis support.

Results and Conclusions

The authors of the Profile followed several principles in developing their OMG proposal including

- Focus on quantifiable notions of time and resource
- Don't inhibit modelers due to analysis
- Must be able to analyze model and predict salient real-time properties
- Modelers should not need deep understanding of analysis techniques

This early examination of using the Profile in support of complex system engineering tasks was encouraging. Clearly, the Profile provides a framework that encourages engineers to be specific, i.e. quantitative, about timing and resource issues. Use of the Profile increases the likelihood of clearer communication of timing and sizing requirements and design given the agreed-to vocabulary backed by tool support. During our experimentation, we found early on that we had misunderstood a description of timing requirements for the configuration we examined. This was exposed once we tried to construct the model and found an obvious area where we had misunderstood the requirements. By the early construction of a model, the mistake was identified and corrected earlier and theoretically at a lower cost. Also, by running analysis using various estimates of load and execution times, we achieved greater insight into potential bottlenecks than would have been revealed in the creation of traditional UML artifacts.

The approach of capturing design using UML class, object, and state diagrams is consistent with how UML is often applied on DoD programs. No UML diagrams were created during this activity that would not have been created during normal development work (normal meaning an activity that is not focused on examining the RT-UML Profile). This is beneficial since designers are more likely to work with analysts if the designers can use the tools and techniques they are already comfortable with.

Use of sequence diagrams to define load is straightforward. An especially attractive aspect of using sequence diagrams in this manner is that the load can be incrementally constructed by the addition of more sequence diagrams. This scales nicely since the complexity of any one sequence diagram can be kept manageable, yet the analysis can be done across one or more of these. We also found we gained insight by examining output from the analysis tool for increasingly loaded scenarios which was achieved by adding sequence diagrams to the analysis load. By comparing outputs, one could quickly see where bottlenecks would be likely.

A goal of the Profile authors was that the technology could be used by non-experts. While it is unrealistic to expect people with no performance engineering experience to be able to use this technology, it was observed during this experiment that the tools greatly facilitated setting up a timing analysis. For example, the analysis tool correctly interpreted UML entity relationships such as the fact that a chain of actions should be treated as a single logical thing (e.g., a distributable thread) for analysis purposes.

A common reason cited for not undertaking performance and timing analysis early in the development phase is the lack of useful timing information. This experiment indicated that there can be important information available early in the development cycle that makes the analysis a worthwhile activity. For

example, timing constraints, typically in the form of deadlines, are often known at system and subsystem levels perhaps as early as contract award where they may be embedded in requirements documents. The same is often true for network or communication channel throughput rates.

For example, the configuration examined had deadlines associated with the processing of each of the components, e.g., signal processing, and tracking. Similarly, it is usually understood early on what the external communication partners, protocols, and capacities will be. This information can be used during the requirements analysis and architectural definition phases to get estimates on the amount of resources needed, where concurrency is needed, and where potential bottlenecks are likely. Even execution times, the area in my experience most cited as lacking in accurate information, can often be estimated based on prior work. In this experiment, execution times were not known, but were varied to allow the effects to be seen on resource utilization rates and response times. Since we were using analytic techniques, quick turnaround could be done on a variety of potential execution times. These estimates could be passed to developers to be used as budgets.

While a stated principle of the Profile authors was to focus on quantitative notions of time and resource, results of this experiment indicate that use of the Profile also helps in giving the engineer better qualitative insight into the temporal characteristics of the system. Traditional development techniques focus on decomposing the problem into chunks that can be worked on by small teams of engineers. Once the chunking has been accomplished, the focus is on the operations and data that have to be used and provided by these chunks. Little of this causes engineers to think about temporal and concurrency challenges and solutions. Using the Profile during this experiment caused us to gain significant insight into the logical chains (distributable threads) of processing that the system naturally broke into as well as where the resource management focus should be (due to contention of resources by multiple distributable threads). This qualitative insight is probably as valuable as the quantitative insight gained.

The tools were found to be useful in their current state although some improvements would be helpful. Allowing post-period deadlines would be useful since it is more difficult to accurately model a case where there is a chain of processing that occurs across a number of UML active objects (possibly deployed on separate nodes) where the total deadline for this chain exceeds the period of the initiating stimuli. For example, consider when AO1 triggers AO2 which triggers AO3. It is not uncommon to have an occurrence pattern for AO1 that is periodic with a period that is smaller than the deadline for completing the chain of AO1 to AO2 to AO3 processing. RapidRMA does not currently directly support this situation although a workaround exists.

Next Steps

The U.S. DoD will use this technology in the development of next generation, large-scale multi-mission C2ISR systems. In the near term, we will work with Tri-Pacific to better understand their analysis algorithm used for examining distributed configurations. We will also explore looking at options that use other techniques than those based on rate and deadline monotonic analysis. This is due in part because it is not clear that the rate and deadline monotonic analysis assumptions are a good match for several resources associated with large-scale distributed airborne platforms. Specifically, much processing is not periodic and execution times for some algorithms can be heavily dependent on the characteristics of the data.

An area where the Profile might be improved is to better support distributable threads such as those defined in Real-Time CORBA 1.2. Given the focus on networked computing, having distributable threads be a more prominent modeling construct will likely be increasingly valuable. An important question not addressed by this activity is whether the UML model information produced during typical U.S. DoD development efforts will prove useful for timing analysis. Questions of accuracy and the level of granularity remain. Finally, MITRE is investigating the feasibility of developing an analysis capability using the profile based on a time/utility scheduling approach.

Acknowledgements
This work was supported by the U.S. Air Force under contract number FA8721-04-C-0001 to the MITRE Corporation.