Extending the DII COE for Real-Time

Lt. Col. Lucie M.J. Robillard
ESC/AWW
Dr. H. Rebecca Callison
The Boeing Company
John Maurer
The MITRE Corp.

The Defense Information Infrastructure Common Operating Environment (DII COE) provides an environment in which common reusable infrastructure and applications across information systems help achieve goals for interoperability. The Department of Defense (DoD) has a vision for extending these ideas for reuse and commonality to improve the effectiveness of systems performing real-time command and control (C2) missions. This article outlines the need and proposed approaches for extending the DII COE with real-time capabilities.

The DII COE originated with a simple observation about C2 systems: certain functions (mapping, track management, and communication interfaces) are fundamental to virtually every C2 system. Yet, these functions are built repeatedly in incompatible ways even when the requirements are the same or vary only slightly between systems. If these common functions could be extracted, implemented as a set of extensible building blocks, and made readily available to system designers, development schedules could be accelerated and substantial savings achieved through software reuse. Moreover, interoperability would be significantly improved if common software was used across systems for common functions. Realizing these benefits is DII COE's goal as stated in [1].

Several DoD systems, notably the Global Command and Control System, Global Command Support System, and Theater Battle Management Core System, utilize the DII COE, with additional systems planning anticipated for the DII COE. They are being connected into a global grid that will include, in addition to C2, sensor systems and weapons platforms. With sensors and weapons intrinsically operating in a real-time arena, and with time-sensitive targets becoming increasingly important, the application of the DII COE concepts to real-time C2 becomes more compelling.

Background

In 1996, at the Air Force Electronic Systems Command (ESC) Hanscom Air Force Base, Integrated Command and Control System (IC2S) planners began to explore the application and viability of DII COE concepts. Since critical IC2S missions were expected to respond to stringent real-time requirements that could not be satisfied by the DII COE, the ESC Commander, Lt. Gen. Ronald T. Kadish, directed that all C2 programs develop a set of requirements for real-time extensions to existing DII COE capabilities. In the spring of 1997, Air Force, Army, and Navy representatives met to discuss the high correlation of real-time requirements across the services. In July 1997, the Air Force, Army, Navy, and Marine Corps jointly petitioned the Defense Information Systems Agency (DISA) to charter a DII COE real-time technical working group (TWG) aimed at developing common requirements and recommendations for potential products to provide real-time capabilities to the DII COE. DISA approved the services' request, and the real-time TWG began meeting in August 1997.

Initial studies, conducted at ESC, highlighted numerous, relevant characteristics of real-time systems, subsequently suggesting that a piece-part approach to assembling real-time components would not be effective. In late 1997, the Air Force designated the Airborne Warning and Control System (AWACS) Program Office as executive agent for the DII COE real-time extensions. The DII COE real-time integrated product team (DII COE RT IPT) embodies that executive authority. Because their missions are so closely related, the real-time TWG and IPT are in continuous coordination, conduct joint meetings, and share data. Both the TWG and IPT enjoy the benefits from the active support and participation of Army, Air Force, Navy, and intelligence community representatives.

The concepts described here are the product of these two groups, working in collaboration with DISA.

Understanding Real-Time

Real-time process is where the computation's validity depends on logical correctness and time-sensitive completion. In a real-time system, the time that an activity[1] takes to complete and deliver results is as important to correctness as, for example, the computation's precision or accuracy. What is important is not how fast the system responds but that it responds predictably at appropriate times. For example, a protocol for synchronizing clocks across a communication network (distributed time service) is required to be accurate, not fast.

Hard real-time applies to activities that must be deterministic; critical activities have deadlines. When this processing fails to meet a deadline, the system has failed. For example, a missile-warning radar fails if the radar processor completes its computation, but is unable to deliver target reports before an incoming missile passes through a designated intercept envelope. The design emphasis when building systems with hard deadlines is to guarantee that all deadlines will be met.

Soft real-time is nondeterministic to the extent that an occasional missed deadline can be tolerated as acceptable degraded performance, not a system failure. The value of completing a soft real-time activity decreases after its deadline has passed, but the rate at which the value decreases differs between activities. The operational procedures for dealing with missed deadlines also vary. For example, systems may:

- choose to complete a late action anyway
- abandon an ongoing computation in favor of beginning the next cycle
- attempt a less complex computation instead, and/or
- begin to shed low priority, non-critical functions in an effort to correct the overload problem in future cycles.

The RT TWG recognizes a requirement for real-time extensions to the DII COE to support systems with both hard and soft real-time requirements.

Three fundamental properties are often cited as keys to building systems in which required events occur on time, every time: priority, pre-emption, and predictability. Tasks are assigned priorities according to a real-time scheduling algorithm under which theoretical scheduling guarantees can be made[2]. A pre-emptive real-time scheduler then grants system resources to the highest priority task that is ready to run, even if it must interrupt — or even starve — lower priority tasks. This real-time scheduler will often use some form of priority inheritance to limit the length of time that low priority tasks can hold shared resources and block higher priority tasks waiting for resource access. These techniques differ from priority assignments and scheduling algorithms used in general purpose computing where fairness to all users and good average response times are the objectives.

Achieving predictability depends on the component parts' design. Components must be designed as independently schedulable entities (tasks or processes) whose precise execution schedule may be determined dynamically at runtime by the real-time scheduler. They cannot be hard-coded to a particular execution schedule. To limit priority inversions, each component should minimize the time it holds any shared resource and/or disables pre-emption by a higher priority task. Components must use techniques that can be provably correct and analyzed for sharing access to resources such as peripherals, networks, and particularly shared data.

Each component also needs to be constructed for inherently predictable timing behavior. In practical terms, this restriction means that real-time applications must avoid the use of programming features with unpredictable timing. The list of unpredictable constructs includes programming approaches such as the use of dynamic allocation of memory from a heap, garbage collection, and dynamic paging of virtual memory[3].

Other Characteristics of the Real-Time Domain

While predictable timing is the defining characteristic of real-time computing, there are other characteristics typical of these systems as well.

Concurrency

To respond effectively to events that occur asynchronously in the environment with which the system interacts, real-time systems are often constructed as collections of concurrently executing tasks and processes. In contrast with the concurrent processing inherent in general purpose computing, where processes compete for resources without interacting in other ways, the tasks and processes of real-time systems cooperate closely to achieve mission objectives.

Reliability and Availability

Since military real-time systems perform activities critical to the success of military missions, they typically have rigorous reliability and availability requirements.

Operation in Harsh Environments

Real-time systems often must operate in extreme environments that are far less accommodating than a typical office or computer facility. These systems are often installed on vehicle platforms, e.g., aircraft, tank, or missile. Environmental conditions can be expected to exert significant space, weight, and power consumption constraints. Also, the hardware often must be designed to withstand environmental stresses such as extremes of temperature, shock, vibration, corrosive atmospheres, poorly conditioned electrical power, and severe electromagnetic fields. These considerations significantly restrict the choice of equipment that can be packaged with the real-time system. As such, the impact on the DII COE configuration, operating in a real-time environment, cannot be overstated.

Vision for a Real-Time Common Operating Environment

The vision of extending DII COE for real-time systems, as depicted in Figure 1, begins with the layered architecture in place for DII COE today.

The DII COE Kernel provides the basic interfaces and functions to be used by standards-based infrastructure components and DII COE-compliant applications to achieve portability between systems. The planned DII COE Configurable RT Kernel[4] extends basic DII COE concepts in two ways:

- to build the foundation of predictable execution on which real-time systems depend. The RT Kernel is hosted only on real-time operating systems (RTOSs) that provide real-time scheduling capabilities, reasonably predictable operating system performance, and the services required for timely execution of real-time tasks and processes.
- because many real-time systems operate with limited computing resources, the RT Kernel is configurable. RT Kernel services are selectable, rather than mandatory, and only those services compatible with the capabilities of the RTOS are provided for each platform. For example, the RT Kernel services for a small RTOS like VxWorks®, which supports only single-process, single-user configurations, would not include services that manage concurrent access by multiple users. The integrator of a DII COE-compliant system tailors the RT Kernel by selecting only those services required for the specific

VxWorks is a registered trademark of Wind River Systems.

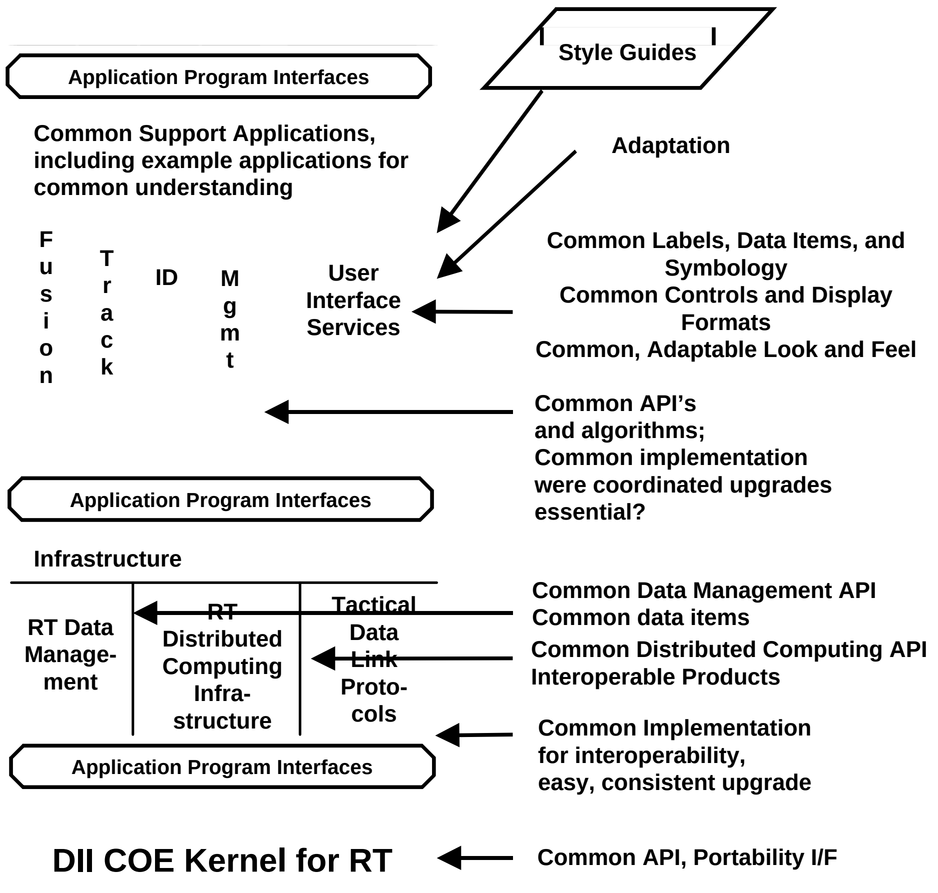
Style Guides
Application Program Interfaces
Common Support Applications, including example applications for common understanding
Adaptation
Fusion
Track
ID
Mgmt
User Interface Services
Common Labels, Data Items, and Symbology
Common Controls and Display Formats
Common, Adaptable Look and Feel
Common API's and algorithms;
Common implementation were coordinated upgrades essential?
Application Program Interfaces
Infrastructure
RT Data Manage-ment
RT Distributed Computing Infra-structure
Tactical Data Link Proto-cols
Common Data Management API
Common data items
Common Distributed Computing API
Interoperable Products
Common Implementation for interoperability, easy, consistent upgrade
Application Program Interfaces
DII COE Kernel for RT
Common API, Portability I/F

Figure 1. *Vision architecture of DII COE for real-time.*

computing configurations of the target system.

Since real-time applications often need a very efficient operating system with small memory footprint for performance reasons, the design philosophy of the DII COE RT Kernel allows a system integrator to tailor the RTOS to meet system needs[5]. Portable Operating System Interface (POSIX®) APIs for operating system services, including APIs for threads and real-time extensions specified in [3], form part of the RT Kernel API. Each DII COE RTOS will be rated for its ability to provide key functional units associated with real-time profiles in the POSIX.13 standard [4]. The dependencies of DII COE RT segments on other DII COE segments and services and on RTOS units of functionality will be documented during the segmentation of a RT software component for the DII COE. A designer of a real-time system can match system requirements to choices of DII COE applications, infrastructure, kernel services, and RTOS.

POSIX is a registered trademark of the Institute of Electrical and Electronics Engineers.

A real-time infrastructure lies above the RT Kernel to provide services for information handling. To support predictable end-to-end execution of system real-time activities, the RT infrastructure must be aware of priority and timing constraints. In the near term, the vision architecture includes a Common Object Request Broker Architecture (CORBA)-based distributed computing infrastructure for real-time. Common implementations of military communications protocols, now available in the DII COE, also must be ported or adapted as necessary for the real-time mission.

It also is envisioned that real-time data management, multi-level trust, and real-time management services will emerge as infrastructure capabilities. Therefore, the infrastructure must be real-time and standards-based to promote interoperability between computing configurations and reuse of applications across systems.

Interoperability and flexibility are key issues at the level of common support applications. Real-time exchange of theater situational awareness is a keystone of emerging concepts for network-centric warfare. The DII COE vision for real-time embraces a common interpretation of information communicated. Beyond the infrastructure components required for basic communication between computers and systems, DII COE is expected to include other applications that contribute to a common understanding of and response to the operational situation: track management, correlation, combat identification, and fusion of sensor data across systems.

Where feasible, components of today's DII COE will migrate to real-time platforms. In other cases, the real-time environment will provide standard access mechanisms through which real-time components may "reach back" to access nonreal-time functions executing on nonreal-time platforms. In other circumstances, new capabilities may be added for the real-time domain.

The Domain of DII COE for Real-Time

The range of real-time systems runs from small, tightly coupled embedded controllers to large-scale data-intensive tracking and control systems. It is reasonable to ask which of these systems should use DII COE for real-time. The overriding requirement for C2 interoperability in real-time drives the initial definition of the domain of DII COE for real-time. Anytime a real-time system needs to interoperate with other computing systems for effective execution of C2, it becomes a candidate for DII COE compliance assessment. When the system's interactions involve time-critical calculations or the exchange of time-critical data (activities that have real-time constraints), DII COE for real-time is a concern.

DII COE compliance in real-time systems, however, does not imply that every computer of a real-time system must achieve compliance in the same

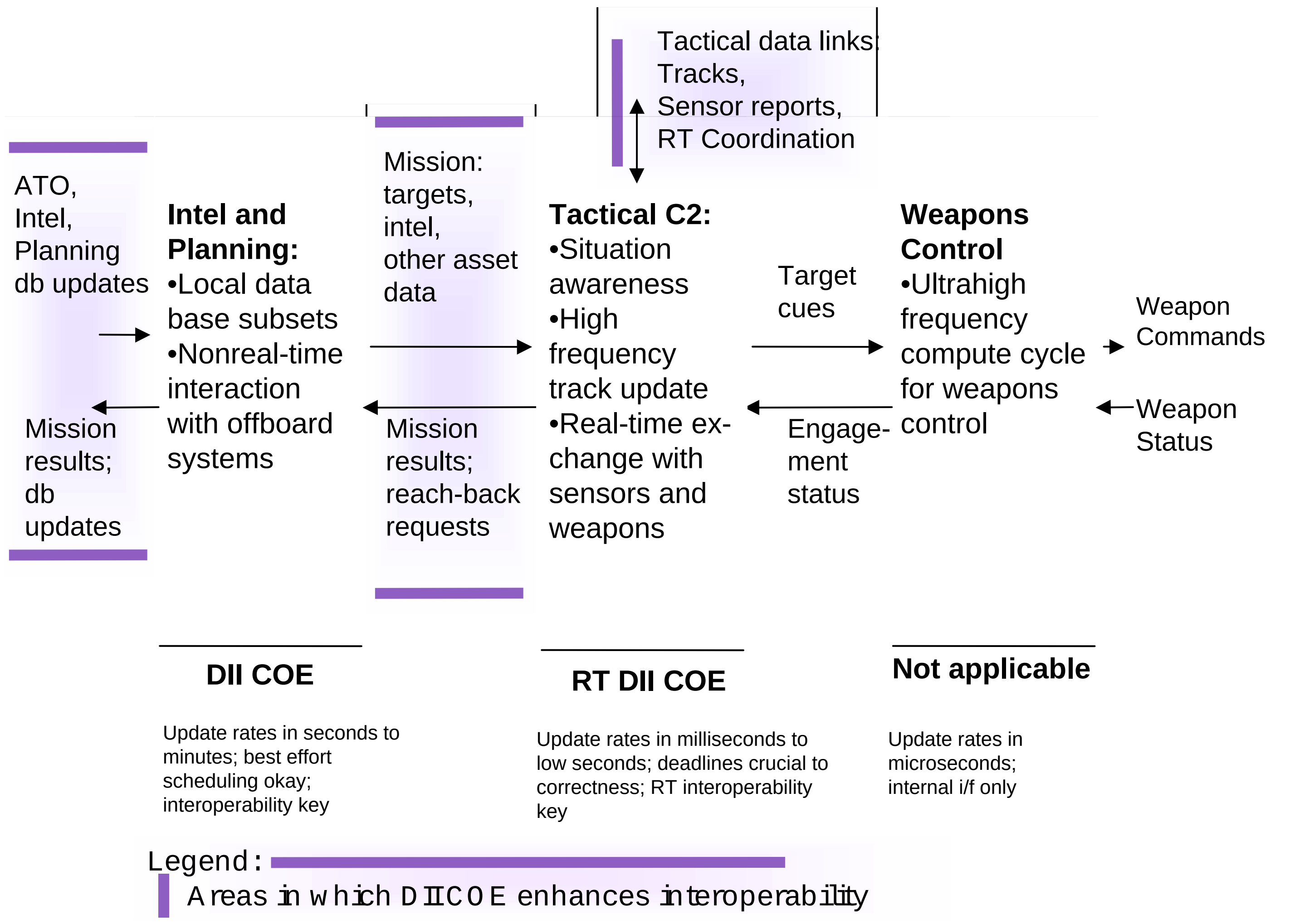
Tactical data links:
Tracks,
Sensor reports,
RT Coordination
ATO,
Intel,
Planning
db updates
Intel and Planning:
•Local data base subsets
•Nonreal-time interaction with offboard systems
Mission:
targets,
intel,
other asset data
Tactical C2:
•Situation awareness
•High frequency track update
•Real-time exchange with sensors and weapons
Target cues
Weapons Control
•Ultrahigh frequency compute cycle for weapons control
Weapon Commands
Weapon Status
Mission results; db updates
Mission results; reach-back requests
Engage-ment status
DII COE
Update rates in seconds to minutes; best effort scheduling okay; interoperability key
RT DII COE
Update rates in milliseconds to low seconds; deadlines crucial to correctness; RT interoperability key
Not applicable
Update rates in microseconds; internal i/f only
Legend:
Areas in which DII COE enhances interoperability

Figure 2. *Breakdown for typical weapons C2 system: DII COE (non-RT), DII COE RT, and exempt.*

way. DII COE-compliant real-time systems will often be comprised of a distributed computing system in which some computers execute the existing nonreal-time DII COE, others are DII COE RT platforms, and exceptions from DII COE compliance are made for others.

Exceptions to DII COE compliance will be made for computing nodes that perform specialized tasks such as signal processing, which typically use specialized hardware and operating systems to achieve their goal. Likewise, DII COE-compliance requirements will not be applied to computer processing units embedded in hardware line replaceable units like network interface cards, controller cards for other computing peripherals, and other controllers that are tightly integrated with commercial or military hardware devices.

Guidelines for assessing how DII COE compliance requirements should be applied in a system are available in [5]. Using this procedure, each computer in a system is uniquely classified as DII COE (non-RT), DII COE RT, or exempt. Figure 2 depicts the allocation that might result in a typical weapons C2 system with external interfaces to other C2 and weapons systems.

Real-Time Capabilities in DII COE 5.0

In DISA Release 5.0, scheduled for October 2000, the following real-time capabilities will be available for incorporation into military C2 platforms:

1. a configurable DII COE RT Kernel for Lynx Operating System (LynxOS™) and Sun Solaris™;
2. a CORBA product with extensions for real-time; and
3. support tools to aid users in developing DII COE RT segments and building customized DII COE-compliant configurations for real-time.

Configurable RT Kernel

As noted earlier, the DII COE Configurable RT Kernel has two parts: an RTOS with POSIX application program interfaces and selectable DII COE Kernel services for real-time.

LynxOS was chosen for the reference implementation for real-time. It provides determinism for hard real-time execution, supports the full range of units of functionality defined in the POSIX.13 standard for real-time profiles, and has the sponsorship of system program offices for several current weapon system development programs. LynxOS provides a solid foundation to support real-time software applications in the DII COE.

LynxOS is a trademark of Lynx Real-time Systems Inc. Sun and Solaris are trademarks of Sun Microsystems Inc.

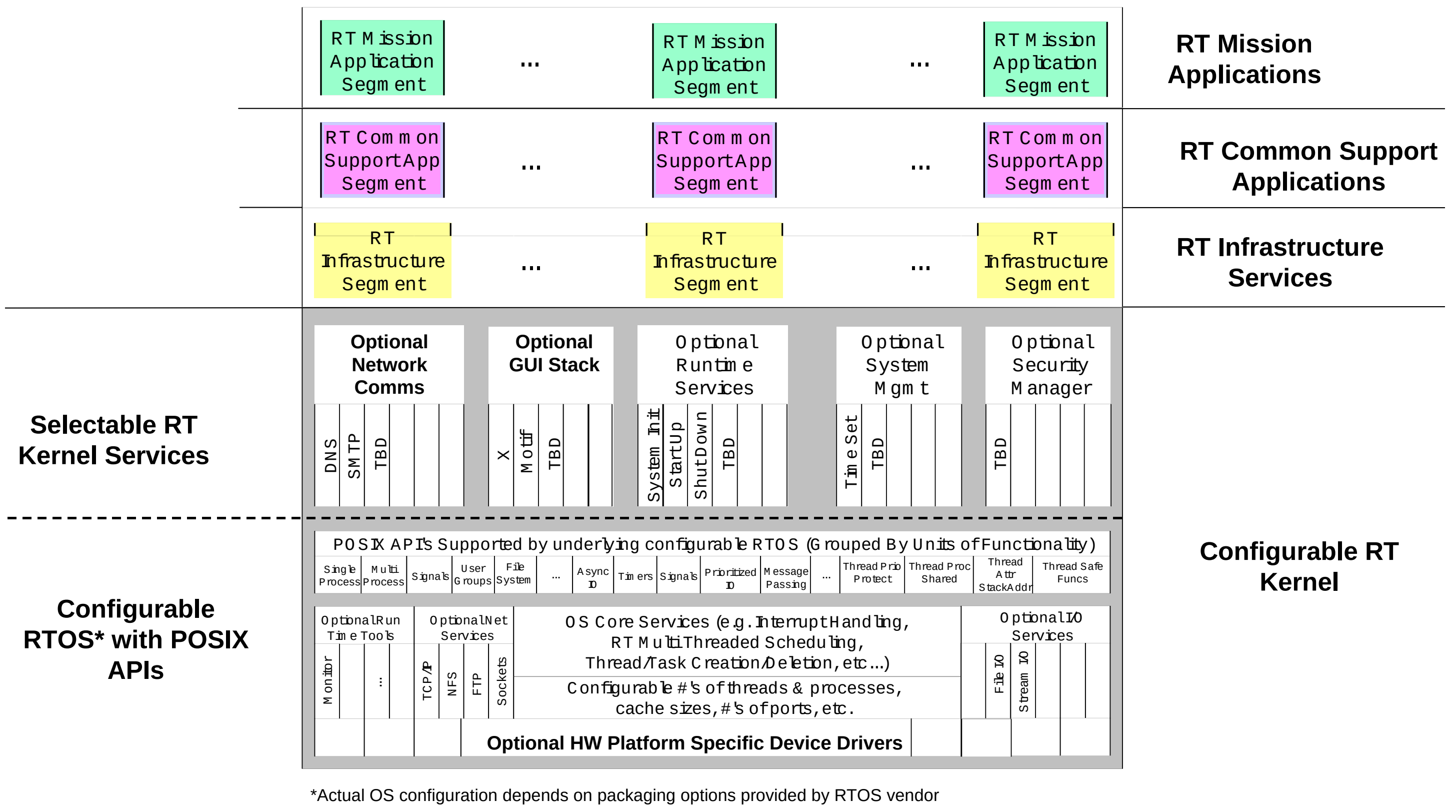
RT Mission Application Segment
...
RT Mission Application Segment
...
RT Mission Application Segment
RT Mission Applications
RT Common Support App Segment
...
RT Common Support App Segment
...
RT Common Support App Segment
RT Common Support Applications
RT Infrastructure Segment
...
RT Infrastructure Segment
...
RT Infrastructure Segment
RT Infrastructure Services
Selectable RT Kernel Services
Optional Network Comms
DNS
SMTP
TBD
Optional GUI Stack
X
Motif
TBD
Optional Runtime Services
System Init
StartUp
ShutDown
TBD
Optional System Mgmt
Time Set
TBD
Optional Security Manager
TBD
Configurable RTOS* with POSIX APIs
Configurable RT Kernel
POSIX API's Supported by underlying configurable RTOS (Grouped By Units of Functionality)
Single Process
Multi Process
Signals
User Groups
File System
...
Async IO
Timers
Signals
Prioritized IO
Message Passing
...
Thread Prio Protect
Thread Proc Shared
Thread Attr StackAddr
Thread Safe Funcs
Optional Run Time Tools
Monitor
...
Optional Net Services
TCP/IP
NFS
FTP
Sockets
OS Core Services (e.g. Interrupt Handling, RT Multi Threaded Scheduling, Thread/Task Creation/Deletion, etc...)
Configurable #'s of threads & processes, cache sizes, #'s of ports, etc.
Optional I/O Services
File I/O
Stream I/O
Optional HW Platform Specific Device Drivers
*Actual OS configuration depends on packaging options provided by RTOS vendor

Figure 3. *Reference architecture for configurable RT Kernel.*

The RT Kernel services to be provided in the initial release 5.0 of DII COE for real-time are 1) commercial off-the-shelf products for X, Motif, and Domain Name Server, and 2) government off-the-shelf services for system startup and shutdown, setting system time, and starting and stopping DII COE processes. These services are documented in [6]. Figure 3 depicts a representative tailoring of the RT Kernel through selection of kernel services and operating system capabilities.

CORBA Infrastructure for Real-Time

CORBA is an international standard [7] for distributed computing that is governed by the object management group (OMG). The CORBA standard provides for flexible interconnection of objects in a client/server model for distributed computing. Four of CORBA's key objectives are support for location independence, operating system independence, hardware independence, and language independence in the design of software components. Figure 4 shows the role that an object request broker (ORB), appropriately extended for real-time, plays in integrating independently developed components into a flexible real-time architecture.

Additions to the CORBA standard to enable real-time computing with end-to-end predictability are documented in the Real-Time CORBA Joint Revised Submission [8], which the OMG is considering adopting. These extensions allow for associating real-time priorities with tasks and requests, passing priority information between communicating components, and the expressing and monitoring of timing constraints for requests. The proposed RT CORBA specification also defines a scheduling service that will provide a consistent real-time scheduling model across a CORBA-based system.

HARDPack by Lockheed-Martin Federal Systems (LMFS) is the leading candidate as the initial RT ORB. HARDPack is a commercial ORB that supports Ada, C, and C++ and includes extensions for real-time performance. HARDPack is cognizant of real-time request priorities and provides the capability to associate deadlines with requests. It extends the CORBA standard with reliable and unreliable broadcast and multicast capabilities, features commonly used for efficient communications in real-time C2 systems. HARDPack also implements the Encapsulated Scheduler to assist in implementing real-time scheduling.

HARDPack is used on at least two Air Force C2 programs: AWACS and Region/Sector Air Operations Center (R/SAOC). Because the RT CORBA standard has not yet been formalized, the real-time extensions to HARDPack are proprietary. LMFS actively participates in standardization and is committed to align its product with the commercial standard within a year of its adoption.

Including CORBA in the DII COE infrastructure for real-time enables the construction of components that can be used in a variety of configurations in different systems. However, it is imperative that components are properly designed to take advantage of this flexibility. When it is reasonable to expect that a given application component will not always execute on the same central processing unit with another component with which it interacts, the component(s) should be designed in such a way that introducing a network between the components can be tolerated[6]. In general, this consideration will drive designs toward relatively large-grained components with coarse-grained

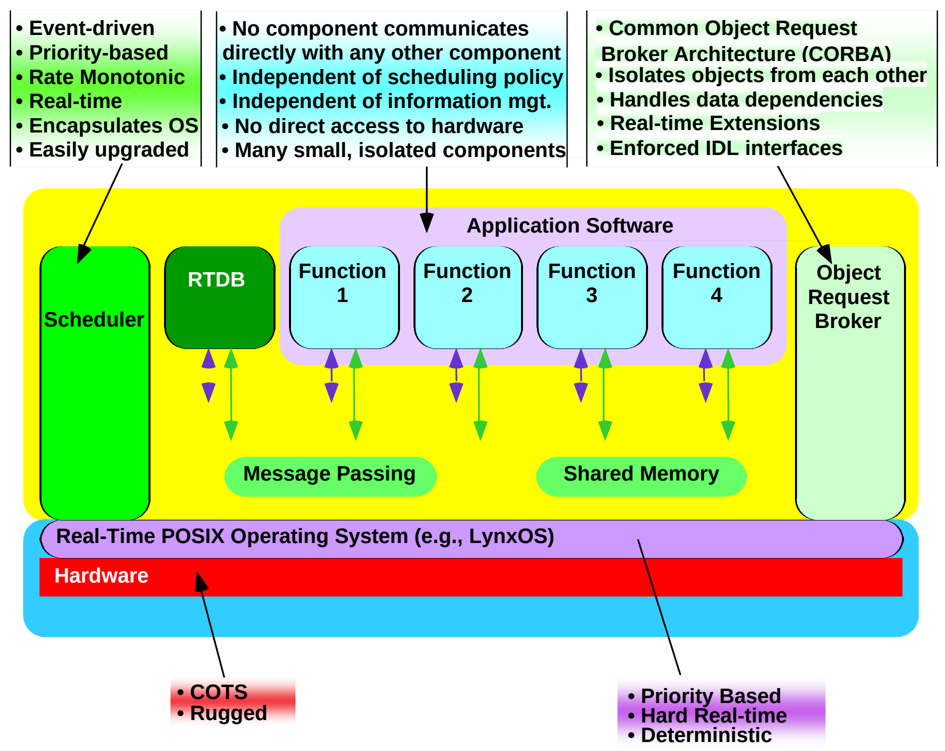
• Event-driven
• Priority-based
• Rate Monotonic
• Real-time
• Encapsulates OS
• Easily upgraded
• No component communicates directly with any other component
• Independent of scheduling policy
• Independent of information mgt.
• No direct access to hardware
• Many small, isolated components
• Common Object Request Broker Architecture (CORBA)
• Isolates objects from each other
• Handles data dependencies
• Real-time Extensions
• Enforced IDL interfaces
Application Software
Scheduler
RTDB
Function 1
Function 2
Function 3
Function 4
Object Request Broker
Message Passing
Shared Memory
Real-Time POSIX Operating System (e.g., LynxOS)
Hardware
• COTS
• Rugged
• Priority Based
• Hard Real-time
• Deterministic

Figure 4. *RT CORBA infrastructure.*

interactions rather than designs involving fine-grained interactions between distributed objects.

Build-Time Integration and Supporting Tools

With a goal of constructing a real-time system with predictable performance and reliable behavior, the integration of real-time segments will take place in the integration laboratory using build-time tools rather than run-time plug-and-play installation. The functions of configuration, link, load, and test in the laboratory will need to be performed using new tools with the DII COE inventory. These Build-Time Tools assist the systems engineer to accomplish several functions in series:

1. select the complete list of DII COE real-time segments for the target environment
2. analyze the inter-segment dependencies
3. choose the selectable RT Kernel Services required by the segments
4. analyze the inter-kernel dependencies
5. and select the required POSIX units of functionality to be provided by the RTOS. The product of the tools is a list of components that must be configured to provide the required functional capabilities. The RT Kernel can then be configured using commercial development tools. The specification for the Build-Time Tools appears in [9].

The Future of DII COE for Real-Time

This effort is on the ground floor to provide a solid foundation on which to build future real-time capabilities in the DII COE. A great deal of work remains to enable widespread reuse of DII COE RT application software; most critical is the establishment of an architecture at the application program interface level. In addition, further requirements analysis and real-time product nominations will be done in the following DII COE functional areas: management services, multi-level trust, mapping, alerts, correlation, message processing, data management, track management, combat identification, and communications. System program offices are being sought out which already utilize software products built with an open architecture and POSIX conformance approach that meet the real-time requirements for C2 interoperability. The goal is to improve C2 interoperability for weapon platforms as the DoD reaches toward the goal of Joint Vision 2010. ◆

About the Authors

Lt. Col. Lucie Robillard is the Air Force Executive Agent for Real-Time DII COE at Hanscom AFB, Mass. She is chairwoman for the DII COE real-time integrated product team (IPT) that has the charter to make real-time extension to DII COE a reality for all services. She is a Level 3 certified acquisition professional. She has joint assignment experience. A majority of her assignments have dealt with software acquisition and engineering. She has a bachelor's degree in electrical engineering from the University of Vermont and a master's degree in systems management from University of Southern California.

ESC/AWW
3 Eglin St.
Hanscom AFB, Mass. 01730
Voice: 781-377-2679
Fax: 781-377-1069
E-mail: robillardl@hanscom.af.mil
Internet: http://spider.osfl.disa.mil/dii/aog_twg/twg/DISAWEB.HTML

Dr. H. Rebecca Callison leads the Boeing team supporting Lt. Col. Robillard and the DII COE real-time IPT. She has 25 years of experience in the design and implementation of real-time systems, principally in the area of defense systems. She has a bachelor's degree from the University of South Carolina, a master's degree in computer science/systems analysis/design from the University of Pennsylvania, and a doctorate degree from the University of Washington. She has served on the faculty of Oregon State University. She has research interests in the areas of software architectures for real-time systems and concurrency control for real-time.

The Boeing Co.
20403 68th Avenue S.
Kent, Wash. 98032
Voice: 253-657-3952
Fax: 253-657-0505
E-mail: rebecca.callison@boeing.com

John Maurer leads MITRE's real-time and performance engineering section. Maurer also chairs the DII COE real-time technical working group. He has a bachelor's degree in mechanical engineering from MIT and 24 years experience implementing software-intensive DoD systems. His work experience includes real-time system development for airborne surveillance systems and Army vehicle systems.

The MITRE Corp.
202 Burlington Road
Bedford, Mass. 01730
Voice: 781-271-2985
Fax: 781-271-4686
E-mail: johnm@mitre.org
Internet: http://spider.osfl.disa.mil/dii/aog_twg/twg/rttwg/rttwg_page.html

References

1. DISA, Defense Information Infrastructure (DII) Common Operating Environment (COE) Baseline Specification, version 3.1, April 29, 1997, DISA Joint Interoperability and Engineering Organization, Reston, Va.
2. Klein, Mark H. et al., *A Practitioner's Handbook for Real Time Analysis*, Kluwer Academic, ISBN 0-7923-9361-9.
3. Information Technology — Portable Operating System Interface (POSIX) Part 1 — System Application Program Interface (API) [C Language], ISO/IEC 9945-1:1996 (E) ANSI/IEEE Std. 1003.1.
4. Draft Standard for Information Technology — Standard Application Environment Profile — POSIX Realtime Application Support (AEP), P1003.13 Draft 9, September 1997.
5. DII COE RT TWG, "DII COE Real-time Decision Tree & Assessment Process: Deciding What's in the Domain," Jan. 20, 1999, http://spider.osfl.disa.mil/dii/aog_twg/twg/RTASSESS.html.
6. DII COE RT TWG, Software Requirements Specification for Kernel Services for the Real-Time Defense Information Infrastructure Common Operating Environment (RT DII COE), (Draft) Revision 1.0, Jan. 8, 1999.
7. Object Management Group, The Common Object Request Broker: Architecture and Specification, Revision 2.2, February 1998. (http://www.omg.org/corba/corbaiiop. html)
8. Object Management Group, Real-Time CORBA 1.0: Joint Revised Submission, Dec.10, 1998. (http://www.omg.org/techprocess/meetings/schedule/Realtime_CORBA1.0_RFP.html)
9. DII COE RT TWG, Build Time Tools Use Case Specification for Real-Time Extensions to the Defense Information Infrastructure Common Operating Environment (RT DII COE), (Draft) Revision 1.0, Jan. 15, 1999.

Notes

1. We use the term "activity" to capture the notion that a time-constrained operation of the system may include computation, communication, and other input/output and may traverse multiple computing nodes.
2. A discussion of real-time scheduling algorithms is outside the scope of this paper. The interested reader is referred to [2] for an overview of available scheduling techniques.
3. The degree to which these techniques must be avoided depends somewhat on the type of real-time system for which the component is intended and the situation in which the feature will be used. If it must perform acceptably in hard real-time systems, the ban on features with loosely bound performance must be strict. If soft real-time is the objective; the restrictions may be relaxed with caution.
4. In the rest of this paper, we will use the term "RT Kernel" as an abbreviation for "DII COE Configurable RT Kernel."
5. Commercial-off-the-shelf (COTS) tools provided by the operating system (OS) vendor are used to configure the OS, not DII COE unique software. The degree to which a specific RTOS can be configured depends on the flexibility the RTOS vendor provides.
6. As in all other situations in which a component is asked to adapt to a new computing environment, it is assumed that developers will not attempt to use the component in any way that violates the laws of physics.