# Composability Problems and Mitigation in Real-Time Software Defined Radio (SCA) Systems

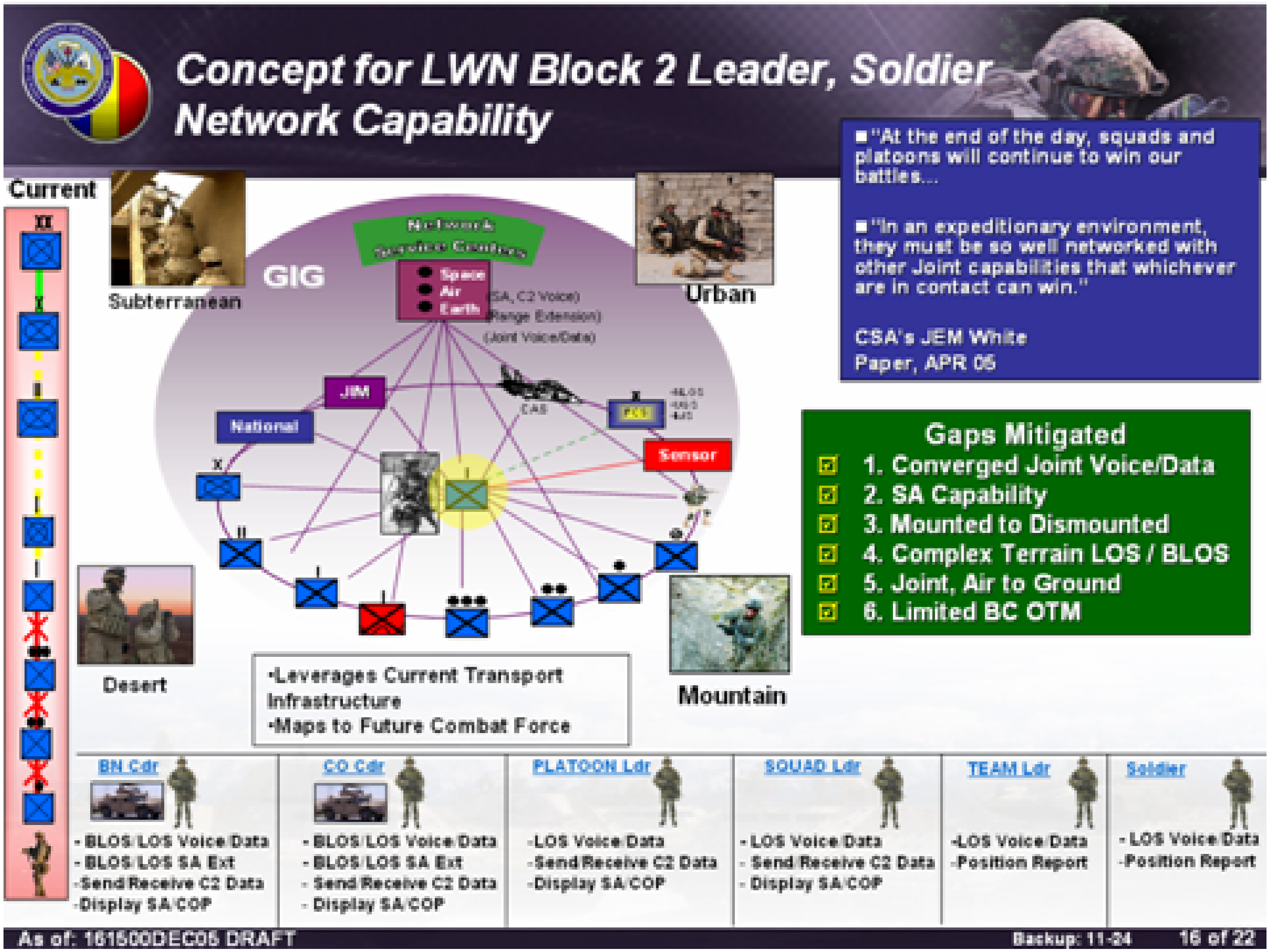

- David F. Carlson, Chronolytics Inc.
- Raymond Lindenmayer, Harris Inc., RFCD


The full length paper is available at:

http://www.chronolytics.com/whitepapers/omg_rt2006.pdf

# Agenda

- Future Situational Awareness: Customer requirements
- SCA Background
- Composability Shortfalls
- Mitigations to Shortfalls
  - Active Objects
  - Rate Monotonic Analysis
  - Deadline Monotonic Analysis
- Recommendation




David Carlson / Ray Lindenmayer
OMG RT 2006

# Background on SCA

- SCA is a collection of CORBA interfaces known as the Core Framework (CF)
- The SCA specifies a POSIX subset known as the AEP
- The functions of the CF are implemented by the Operating Environment (OE)
- The radio applications designed to the CF are known as Waveforms (WF)

# Goals of SCA

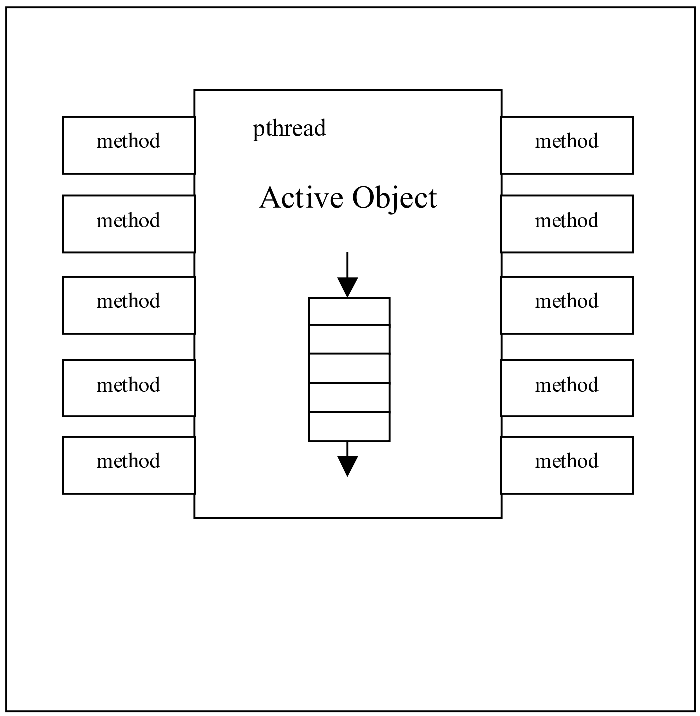
- Ease of integration
- Standardize WFs so that investment can be amortized and interoperability ensured
- Separate applications (WF) from the platform such that low-end single WF or high-end multi-WF platforms can exist

# Problem Statement: SCA is Not (Currently) a Component Architecture

- Component architecture enables increased SW developer productivity by providing function with the otherwise required “global knowledge” of the CF and OE
- Current SCA has limited sense of “capacity” of shared resources
- Synchronous CORBA interfaces (SMI) suggest a conventional “thread & mutex” implementation
- “thread & mutex” widely recognized throughout the OO and RT communities as non-scalable and non-composable
- No deadline/latency contracts

# Mitigation for Composability

- Active Object (AO) is a common design pattern used to improve composability



# Limitations to Active Object

- Still based on “global” priority.  Weak latency guarantees.
- Blocking SMI CORBA in CF effectively prevents latency guarantees
- Pthread per-object fails to scale down to “lightweight” objects
- Fails to compose with multiple / ordered protected objects (nested locks)

# Mini RMA Tutorial

- Tasks (tasklets) must be non-blocking
- Tasklets run at fixed priority proportional to their utilization
- Context switching is “zero” overhead
- Schedulability of n-tasks:

$$U(n) = n(2^{1/n} - 1)$$

- Which converges for large n to ln(2) or 69%
- For a schedulable set of tasks, RMA is known to be optimal (for fixed priority scheduling)

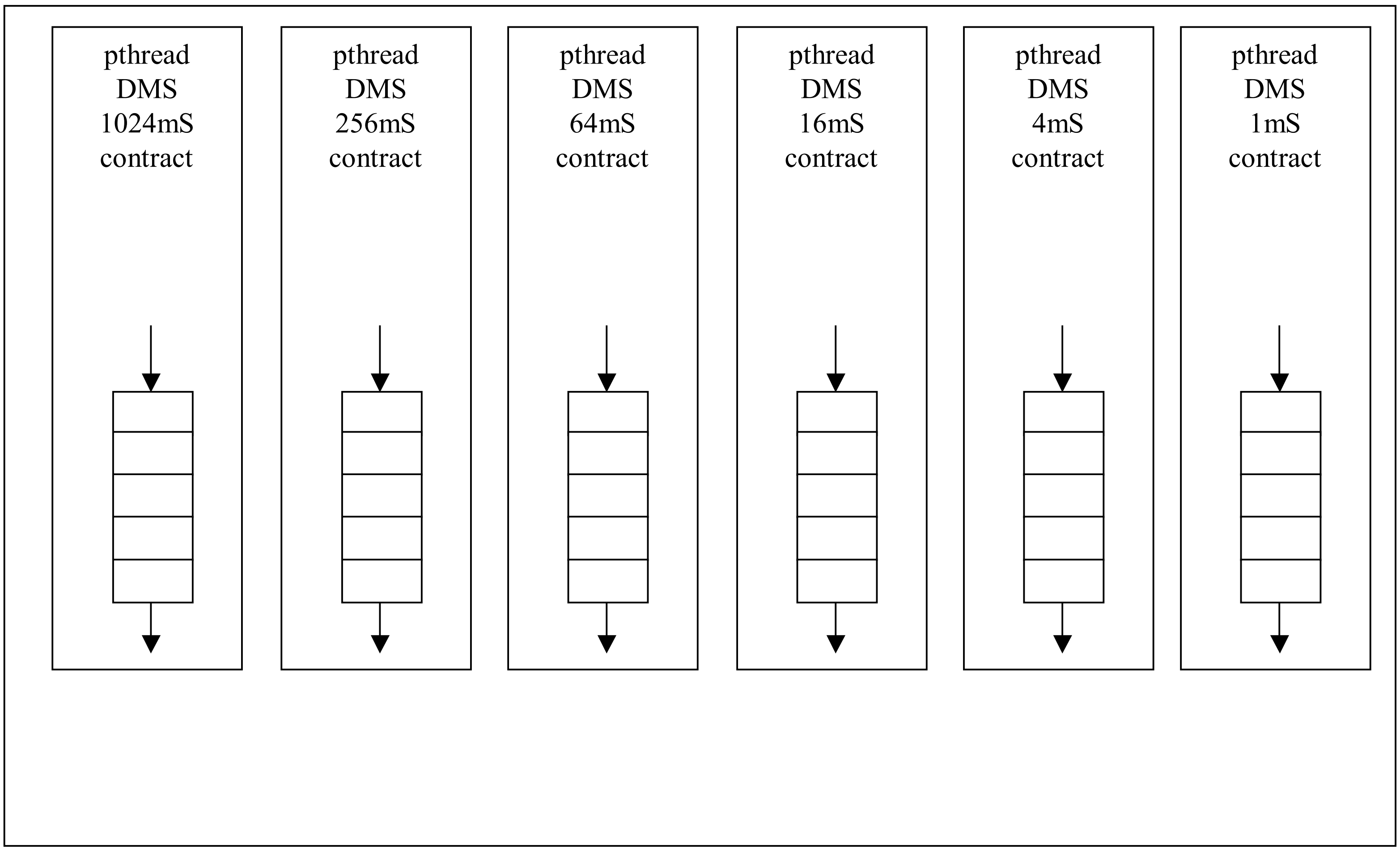
# Rate Monotonic Alternatives to "Thread & Mutex"

- Simulation community has used RMA frequency-based scheduling since the early 90's
- In the early 90's RMA extended to sporadic tasks – Deadline Monotonic Scheduling
- DMS used very successfully within the RT control community (vis., large printing equipment)

# Advantages of Deadline Monotonic Scheduling

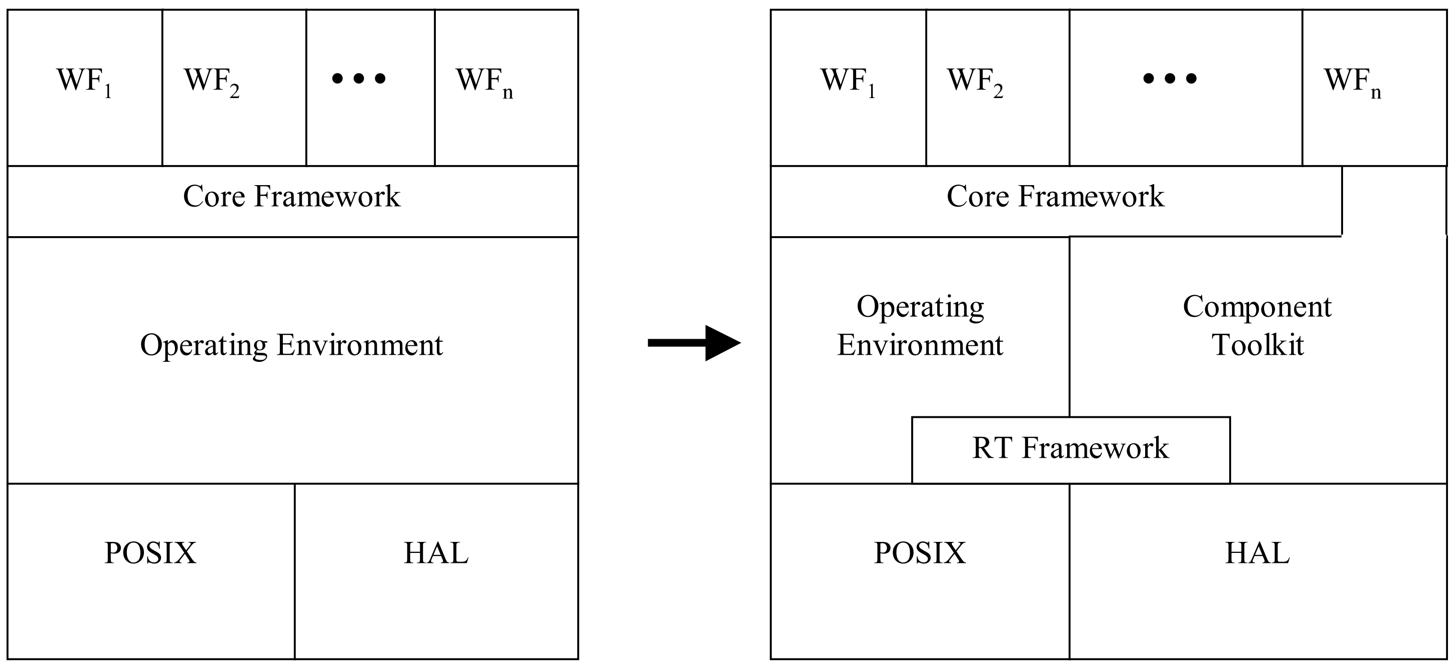
- Facilitates “design for latency”
- SW requirements for real-time operation are allocated from system requirements (Tracability)
- Active Objects are used as “anonymous” latency contract servers
- Specification of RT components possible based on deadline/latency and utilization

# DMS Scheduling Contracts




David Carlson / Ray Lindenmayer
OMG RT 2006

# DMS Advantages to Software Defined Radios



# DMS Advantages to Software Defined Radios

- Integration largely superposition of components
- RT “owned” by framework – performance/metrics centralized. Overruns/failures are tracked
- Framework permits components design of reusable toolkit rather than integration time tweaks of OE thread model
- Apps written to CF/Toolkit -- abstracted from “global” OE/HAL/POSIX
- DMS provides lingua-franca for defining real-time components

# Next Steps

- Improve RT ORB features that support Asynchronous Messaging (AMI)
- Get involved with setting direction for future SCA updates that will allow the flexibility of (if not require) RMA-based solutions