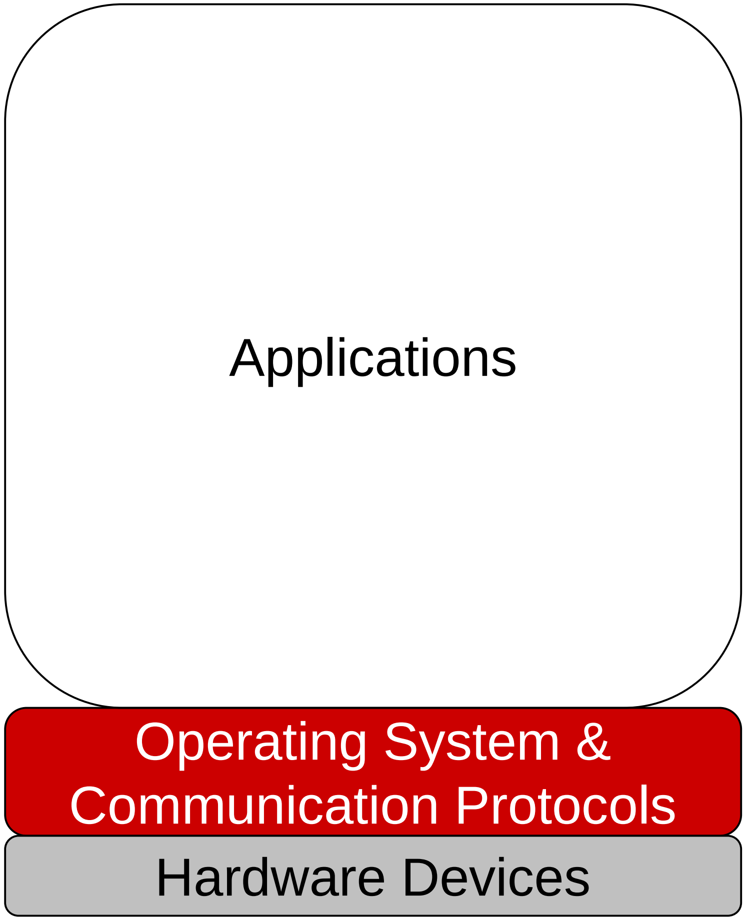
# *Applying Model-Driven Engineering to Evaluate the QoS of Distributed, Real-time, Embedded Systems*

**John M. Slaby**, James Hill, Sumant Tambe,
Dr. Douglas Schmidt

OMG's Workshop on Distributed Object Computing for Real-time and Embedded Systems

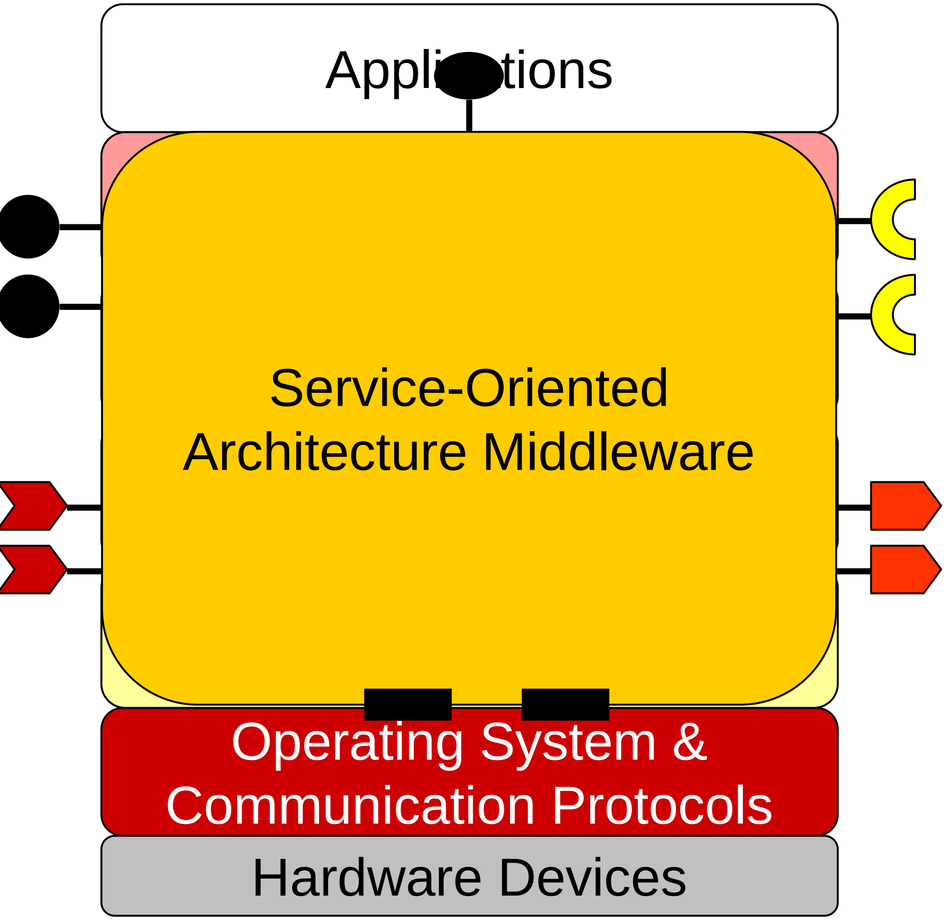
July 10-13, 2006 - Arlington, VA USA

# Motivation: Service-Oriented Architectures




- Historically, distributed real-time & embedded (DRE) systems were built directly atop OS & protocols

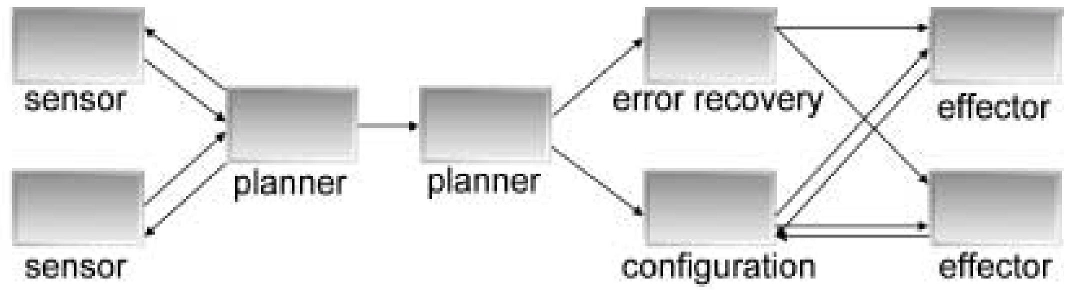
# Motivation: Service-Oriented Architectures



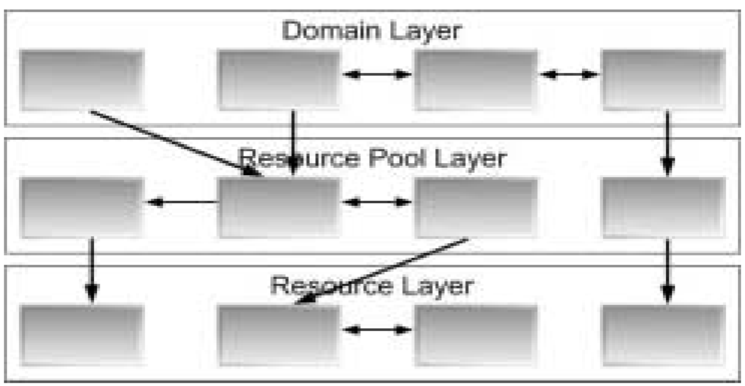

- Historically, distributed real-time & embedded (DRE) systems were built directly atop OS & protocols
- Traditional methods of development have been replaced by middleware layers to reuse architectures & code
  - Viewed externally as *Service-Oriented Architecture (SOA) Middleware*

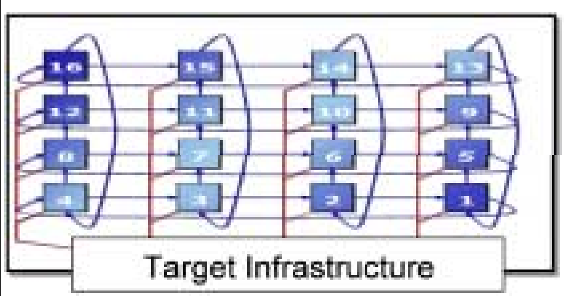
# Serialized Phasing in Large-scale Systems

- Infrastructure developed
- Applications developed
- System integrated




Level of Abstraction

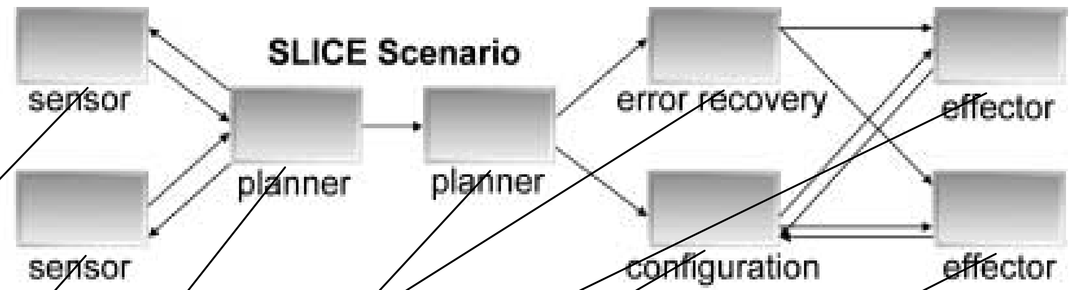


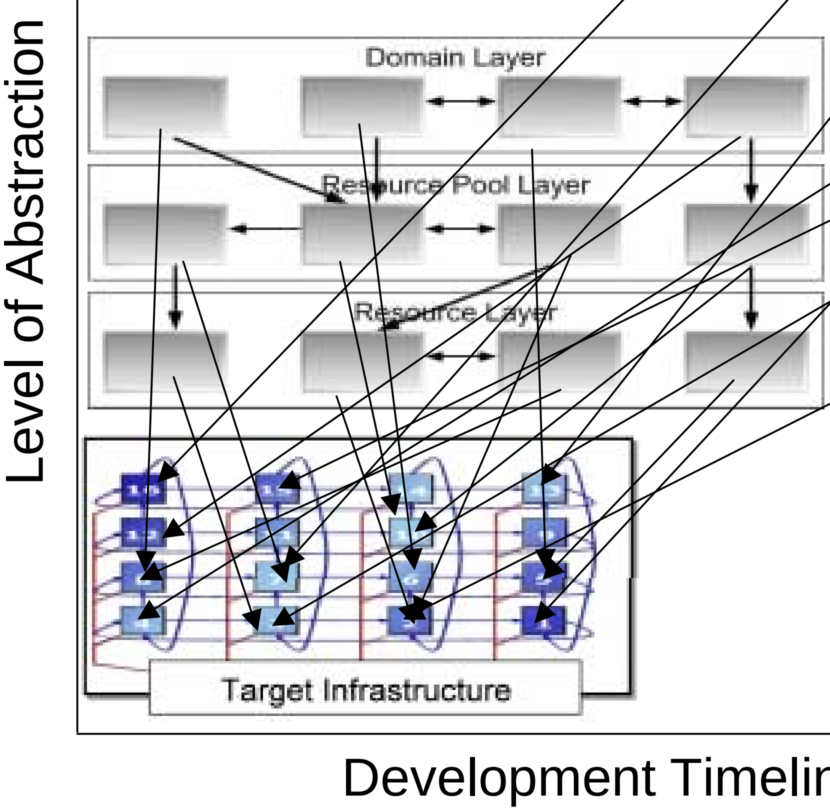




Development Timeline

# Serialized Phasing in Large-scale Systems

- Infrastructure developed
- Applications developed
- System integrated



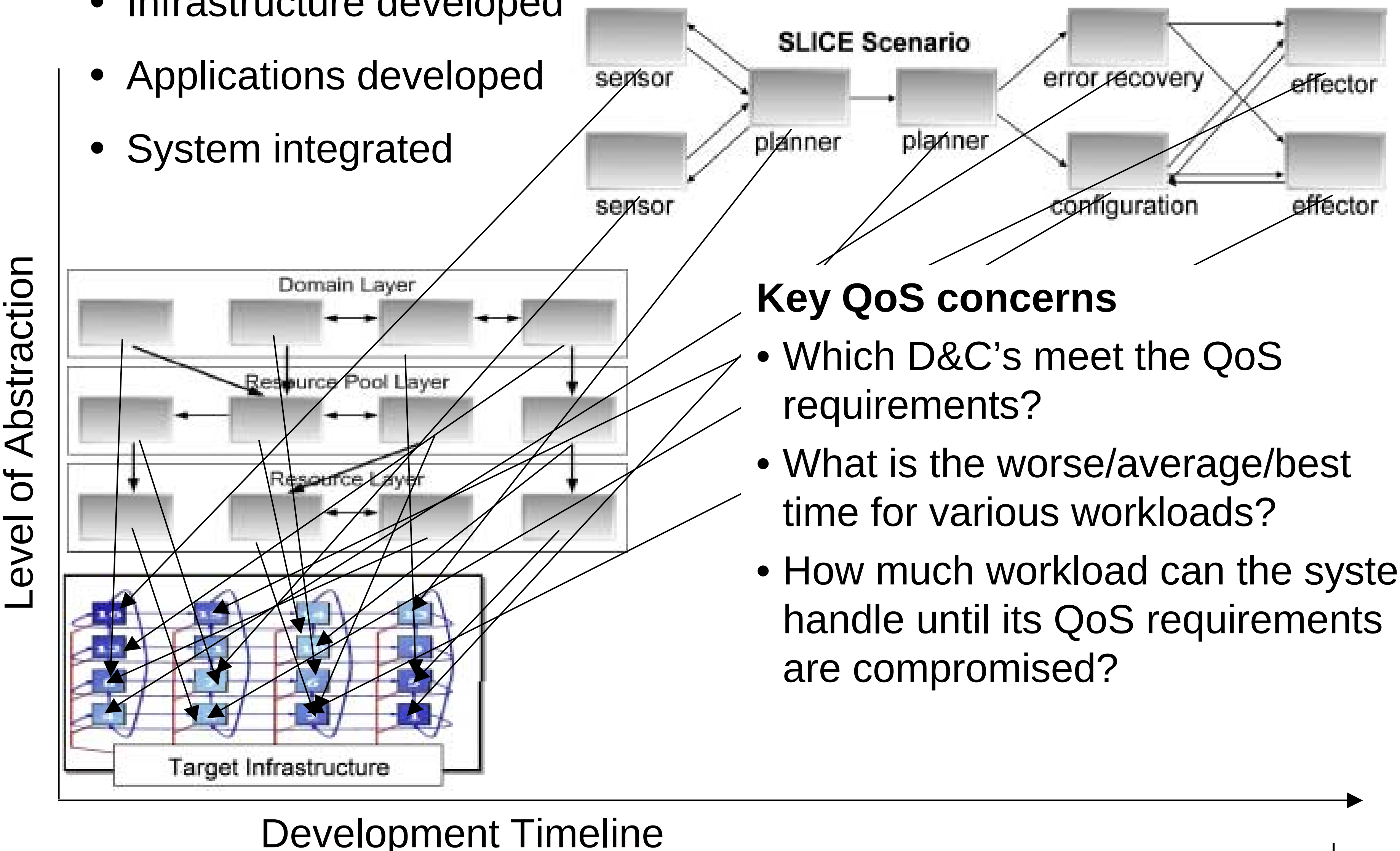




**Complexities**

- System infrastructure cannot be tested adequately until applications are done
- Entire system must be deployed & configured properly to meet QoS requirements
- Existing evaluation tools do not support “what if” evaluation

# Serialized Phasing in Large-scale Systems

- Infrastructure developed
- Applications developed
- System integrated

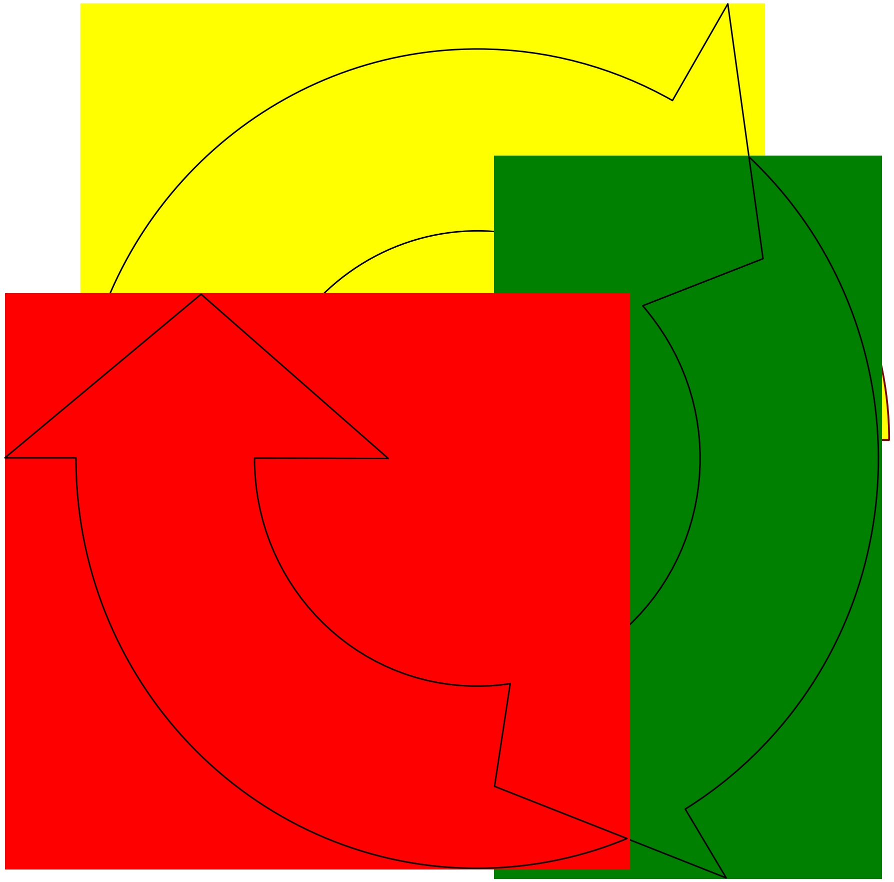



**Key QoS concerns**

- Which D&C's meet the QoS requirements?
- What is the worse/average/best time for various workloads?
- How much workload can the system handle until its QoS requirements are compromised?

# Our Solution: New Generation of System Execution Modeling Tools

- A Toolset which enables early testing and evaluation of SOA-based distributed, real-time applications
  - Modeling tools to build complex distributed applications
  - A component-based workload emulator which can be made to look and behave as an arbitrary SOA application component
  - A test harness to capture and analyze performance data

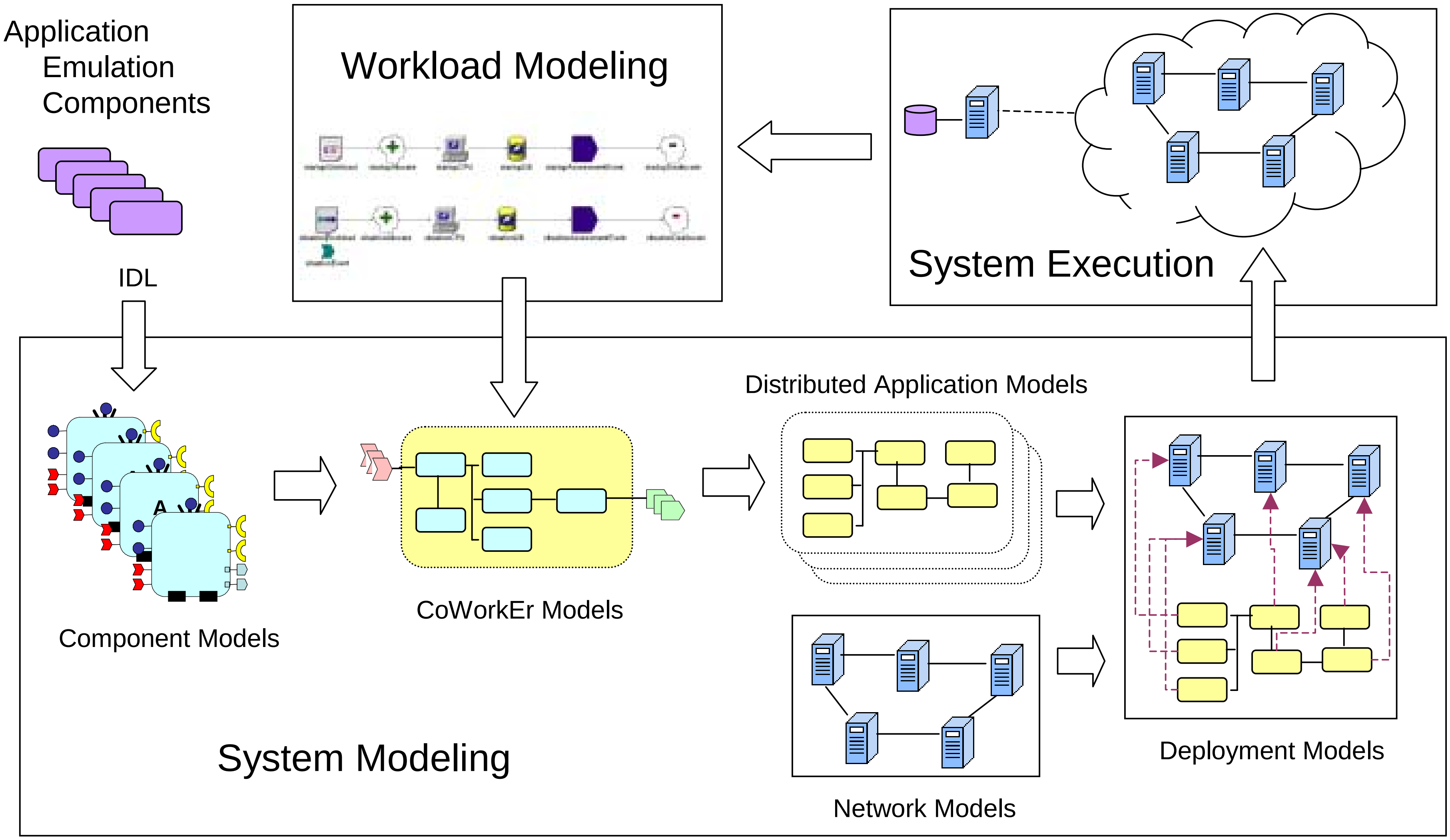


# The System Execution Modeling Toolset

- CCM – CORBA Component Model (SOA Middleware)
- GME – Generic Modeling Environment – for Domain-Specific Modeling Language (DSML) development
- PICML – Platform-Independent Component Model DSML

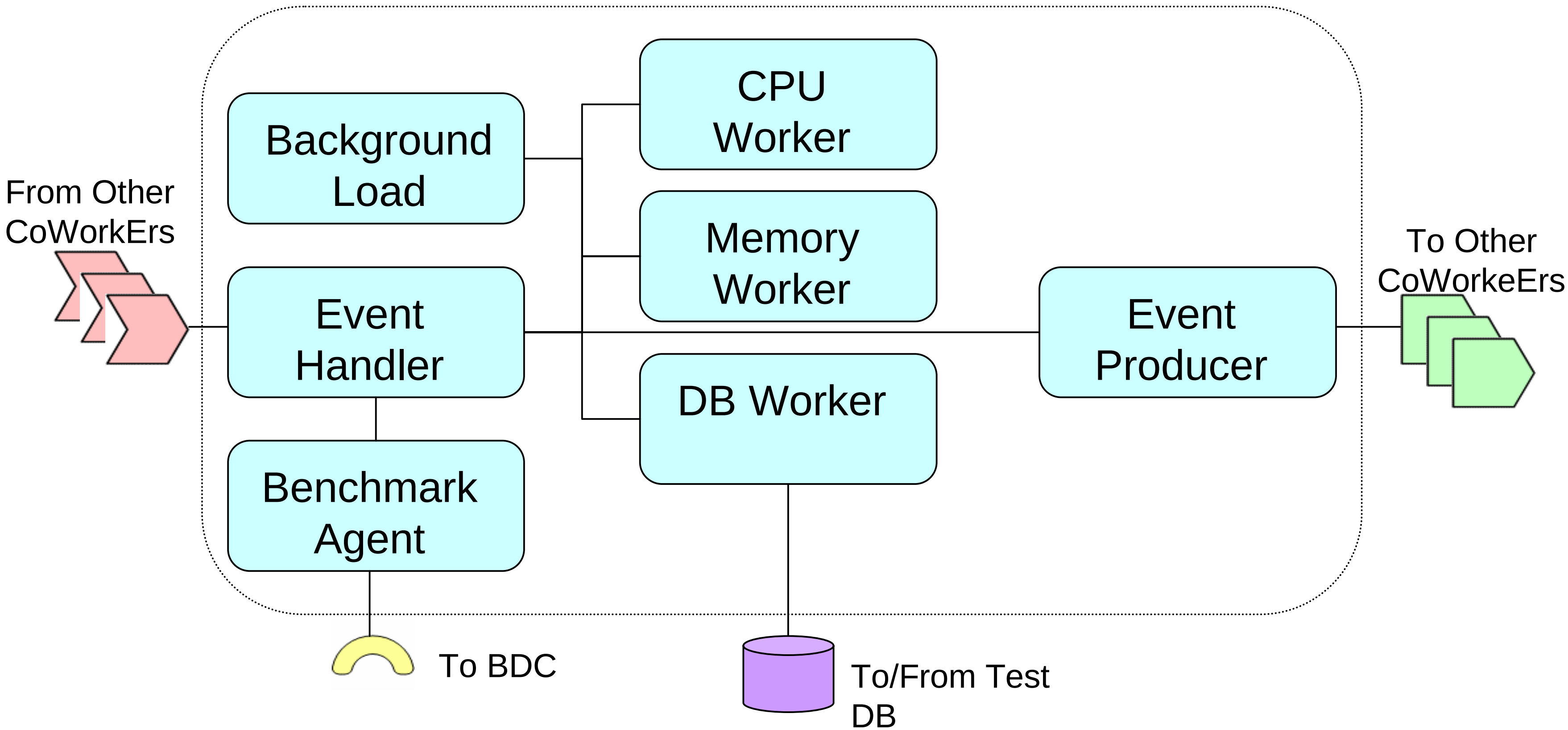
- *CUTS  - CoWorkEr Utilization Test Suite*
  - CoWorkEr – Component Workload Emulator – CCM Component capable of emulating an arbitrary application component
  - WML – Workload Modeling Language - DSML for characterizing behavior of CoWorkErs
  - BDC – Benchmark Data Collector
  - BMW – Benchmark Manager Web – browser and web service interface to test control and test results

# The Big Picture



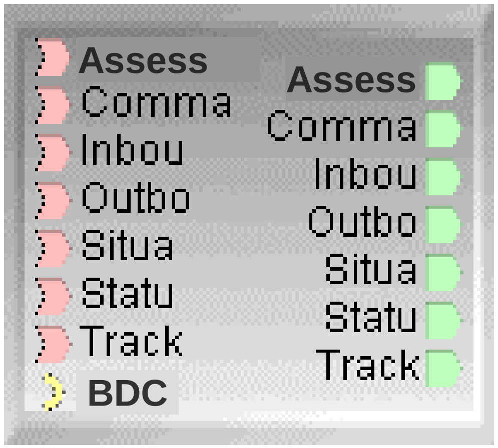
# Building Test Scenarios

# The Component Workload Emulator

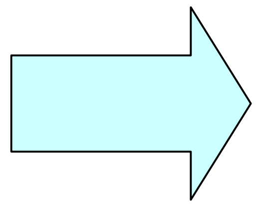


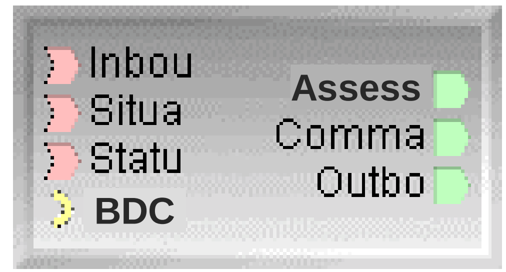

The CoWorkEr is comprised of a set of generic capabilities representing the behavior of an application – CPU load, memory use, database interactions, background load, and event publications and subscriptions

# CoWorkEr Interface Refinement




Generic CoWorkEr Assembly

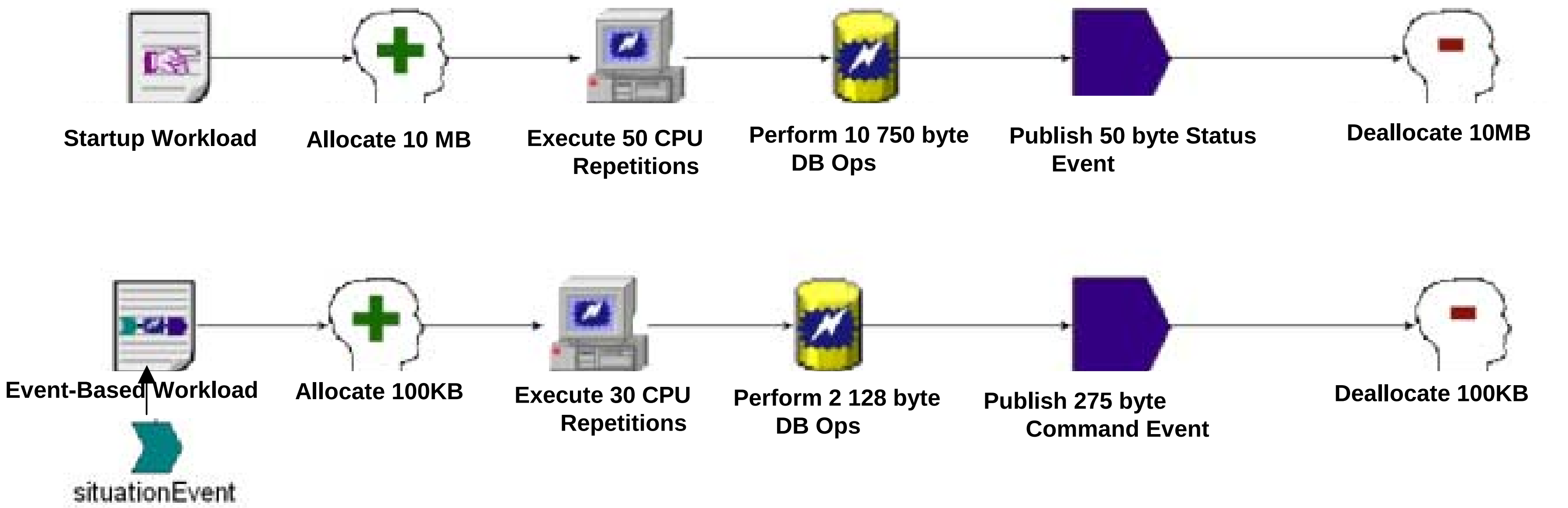





Plan-1

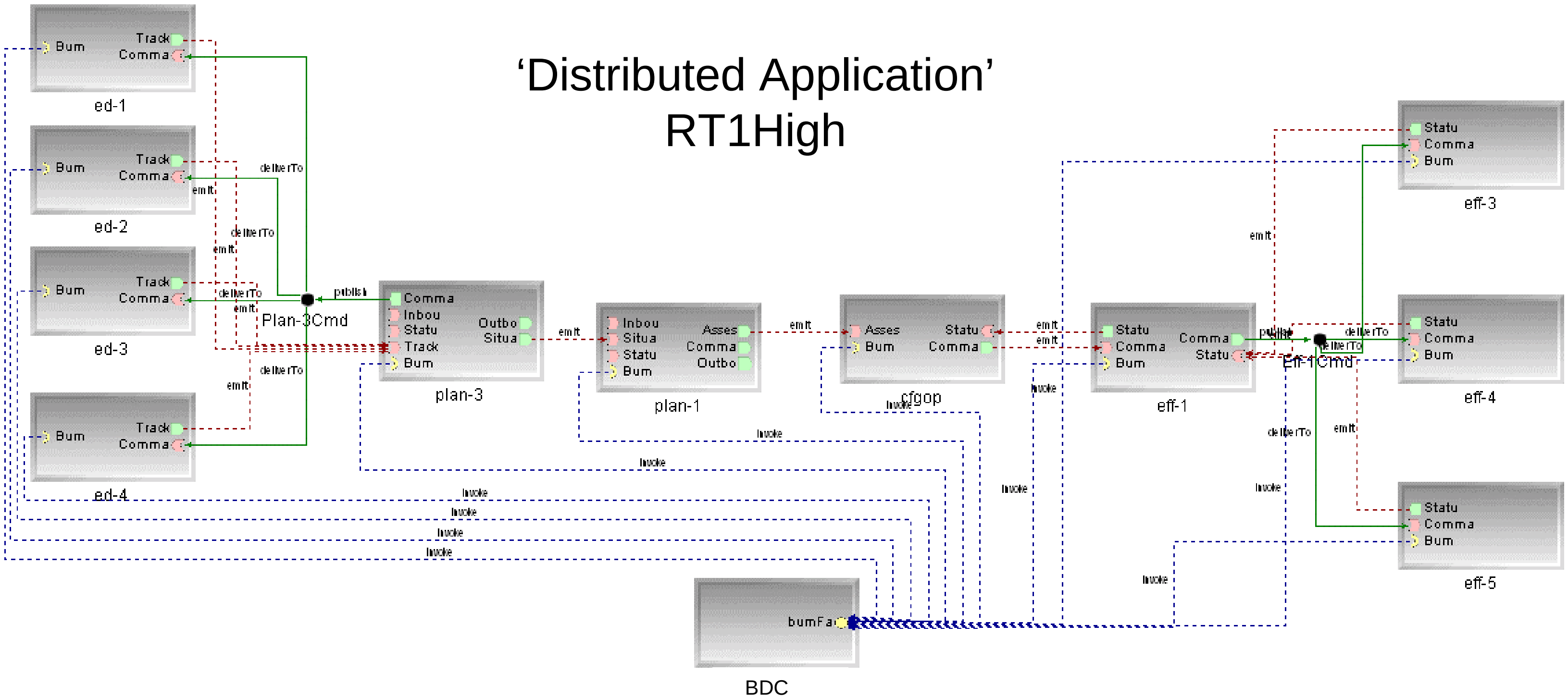
PICML is used to refine the interfaces – transforming the generic CoWorkEr into a component that looks **from an interface perspective** like the application it is emulating

# Workload Characterization




The Workload Modeling Language (WLM) is used to describe behavior – transforming the generic CoWorkEr into an application that looks **from a resource consumption and message handling perspective** like the application it is emulating

# SOA Application Emulation




A test scenario is created by building an assembly of these characterized CoWorkErs. PICML and WML are used to model the scenario, and CCM is used to deploy the CoWorkErs based on those models.

# Test Metrics and Analysis

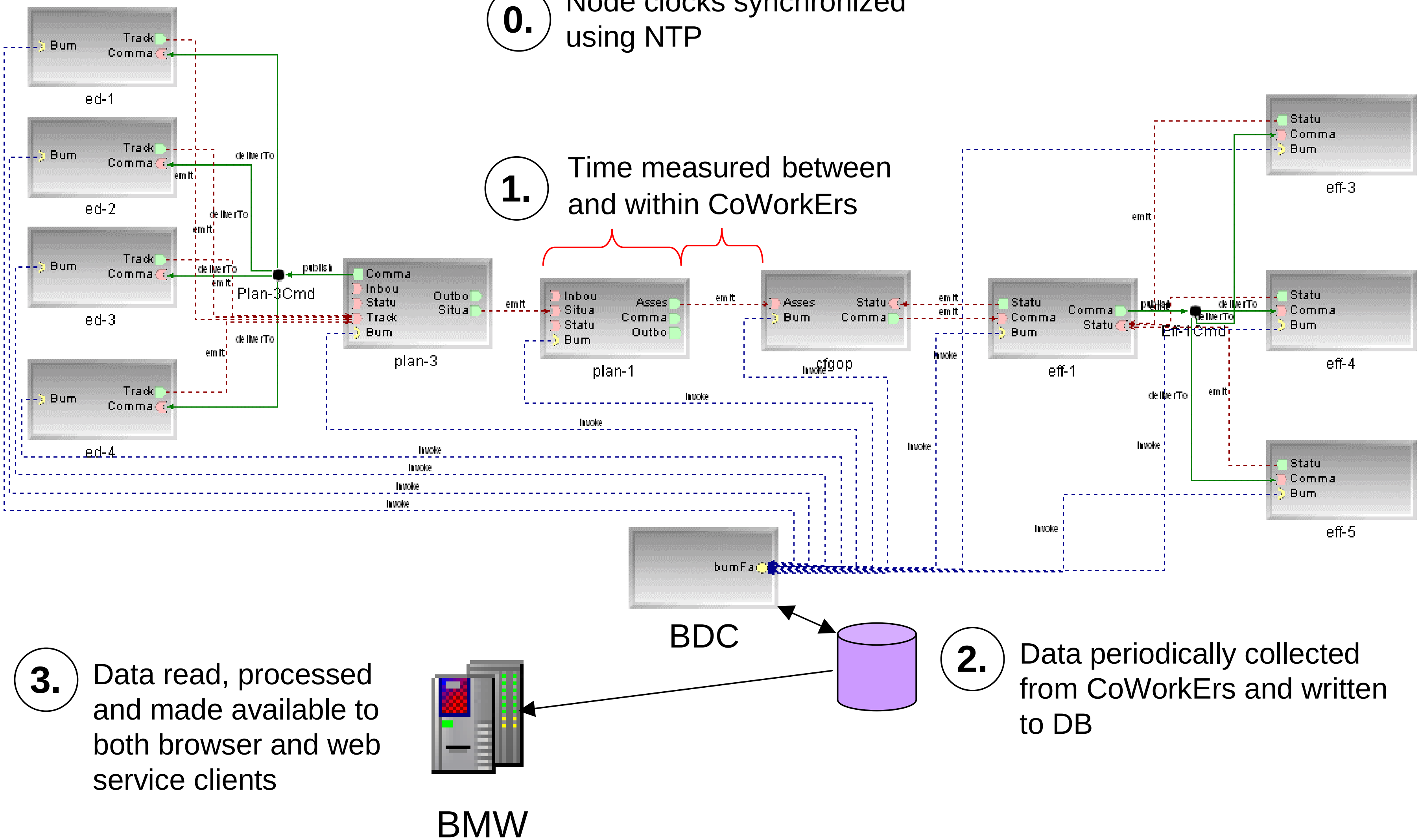
# Timing Measurements

0. Node clocks synchronized using NTP

1. Time measured between and within CoWorkErs

2. Data periodically collected from CoWorkErs and written to DB

3. Data read, processed and made available to both browser and web service clients



# Timing Measurements

http://l71035920/WLGBenchmark/TestResults.aspx?ItemID=246

## Workload Generator Timing Data for Test 246

Return

**Deployment: 'RT1A 3 Node'**

| Critical Path | Deadline (ms) | Analysis |
|---|---|---|
| RT1A | 800 | |

| Host | Event -> | WLG | Workload | Timeline | Snapshots | Avg Samples* | Avg/Rep (ms)* | Average (ms) | Worst Case (ms) | Best Case (ms) |
|---|---|---|---|---|---|---|---|---|---|---|
| L71035919.ssd.ray.com | Status | Effector-1 | Transmission Delay | | 239 | 3 | | 186.3 | 4637 | 0 |
| | | Effector-1 | Total Workload | | 239 | 3 | | 14.64 | 161 | 0 |
| | | | - CPU | | | | 0.37601 | 3.76 | 151 | 0 |
| | | | - Database | | | | 0.5318 | 5.32 | 80 | 0 |
| | | | - Memory | | | | 0.00418 | 0.08 | 10 | 0 |
| L71035919.ssd.ray.com | Command | Effector-1 | Transmission Delay | | 240 | 120 | | 163.42 | 5762 | 0 |
| | | Effector-1 | Total Workload | | 240 | 120 | | 121.49 | 891 | 20 |
| | | | - CPU | | | | 0.77807 | 38.9 | 261 | 10 |
| | | | - Database | | | | 0.64884 | 16.22 | 140 | 10 |
| | | | - Memory | | | | 0.00398 | 0.4 | 100 | 0 |
| | | | - Publication | | | | 53.68565 | 53.69 | 540 | 0 |
| L71035919.ssd.ray.com | Command | Effector-3 | Transmission Delay | | 240 | 120 | | 55.31 | 510 | 0 |
| | | Effector-3 | Total Workload | | 240 | 120 | | 51.03 | 411 | 20 |
| | | | - CPU | | | | 0.71697 | 35.85 | 401 | 10 |
| | | | - Database | | | | 0.57175 | 14.29 | 170 | 10 |
| | | | - Memory | | | | 0.00351 | 0.35 | 70 | 0 |
| L71035919.ssd.ray.com | Command | Effector-4 | Transmission Delay | | 240 | 120 | | 62.16 | 871 | 0 |
| | | Effector-4 | Total Workload | | 240 | 120 | | 105.55 | 641 | 40 |
| | | | - CPU | | | | 0.88025 | 88.03 | 631 | 30 |
| | | | - Database | | | | 0.62082 | 15.52 | 130 | 10 |
| | | | - Memory | | | | 0.00332 | 0.66 | 90 | 0 |
| L71035919.ssd.ray.com | Command | Effector-5 | Transmission Delay | | 240 | 62 | | 218.38 | 922 | 0 |
| | | Effector-5 | Total Workload | | 240 | 62 | | 486.26 | 2323 | 390 |
| | | | - CPU | | | | 0.48752 | 243.78 | 1993 | 170 |

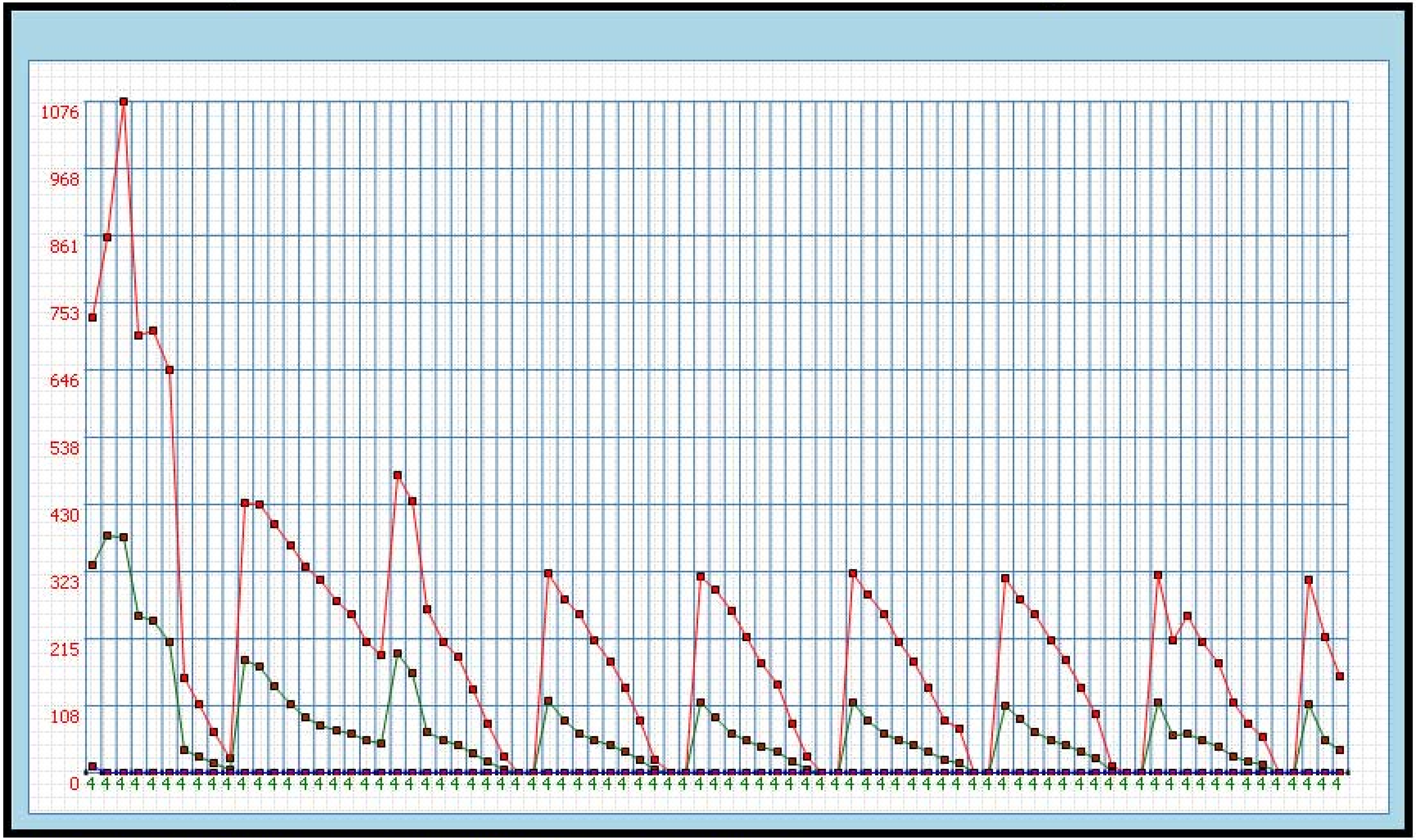
1 2 3

*Note 1: 'Avg Samples' = Average samples per snapshot; 'Avg/Rep (ms)' = Average time for a single worker repetition to execute*

*Note 2: 'Transmission delay' is measured from time event is transmitted until the event is pulled from the destination queue. NTP synchronization between nodes is*

# Timing Variation Analysis

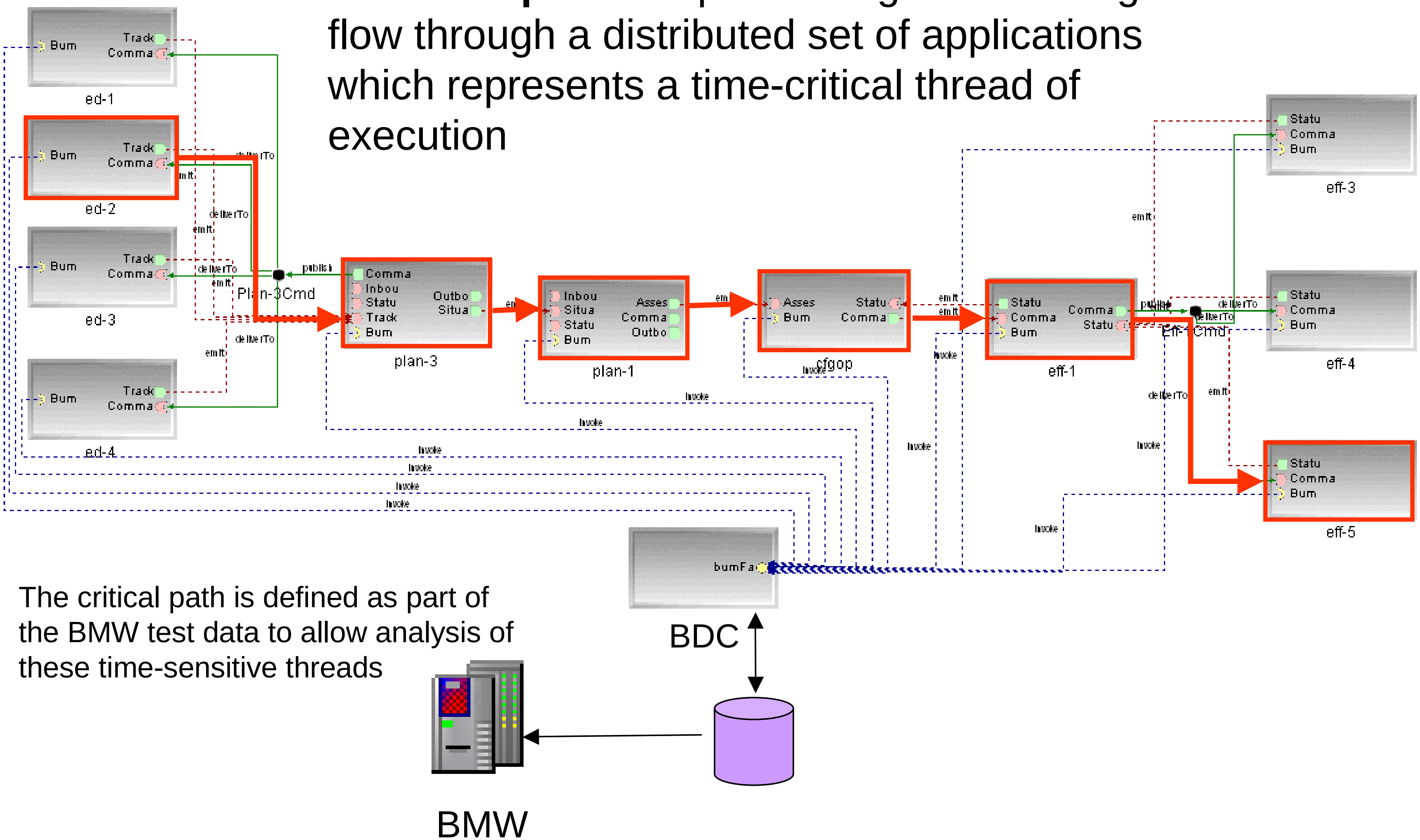
## Timing Variation for Workload in 'Plan-3' Resulting from 'Track'




Legend: X-Axis: number of events per sample, Y-Axis: time in milliseconds
Red: worst case timing per sample, Green: average time per sample, Blue: best case timing per sample

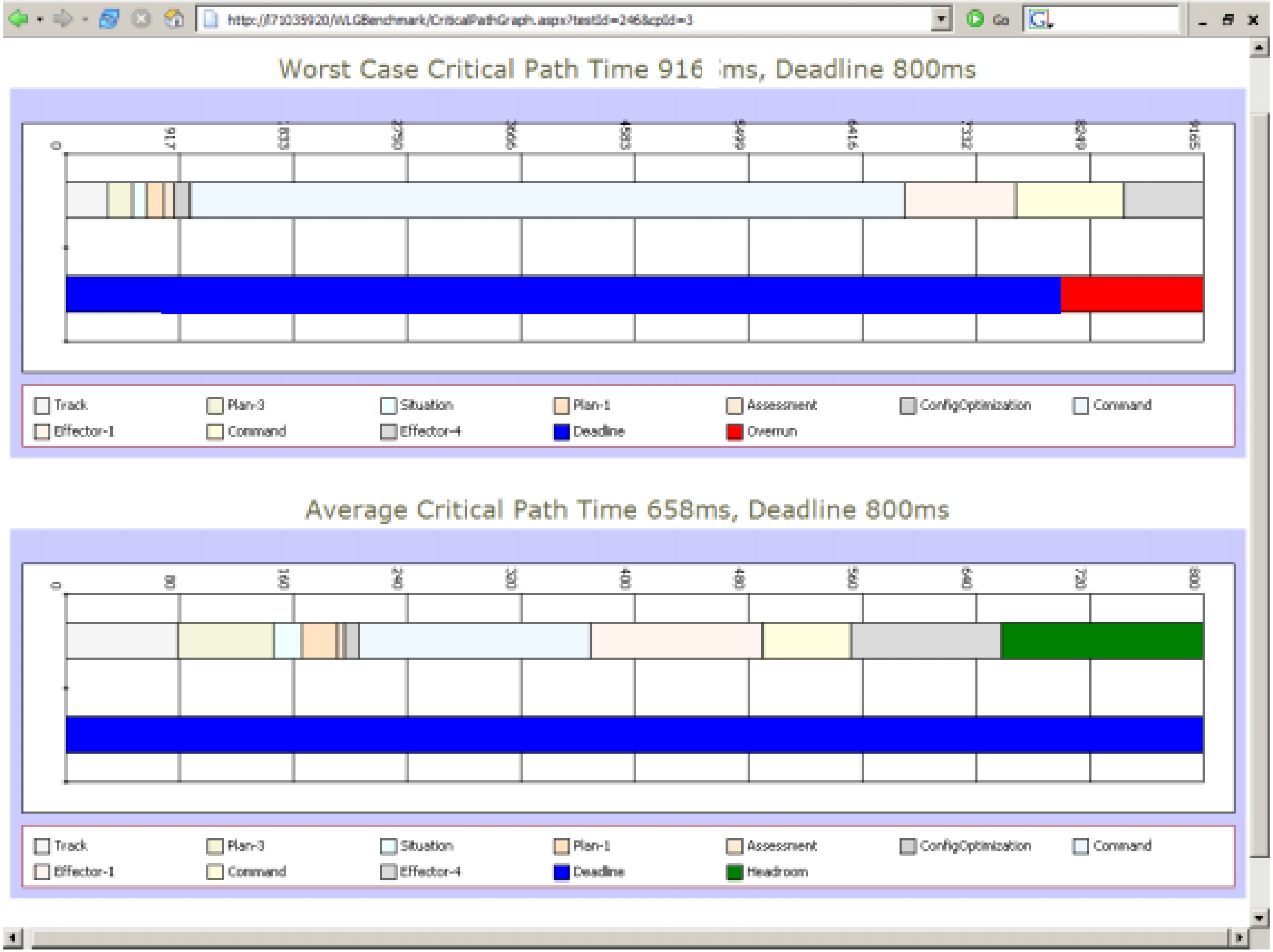
# Critical Path Analysis

The **critical path** is a processing and message flow through a distributed set of applications which represents a time-critical thread of execution




The critical path is defined as part of the BMW test data to allow analysis of these time-sensitive threads

# Critical Path Analysis



# Summary

- Some of the biggest challenges in effective use of these tools involve getting reasonable values for workload. Engineers are generally reluctant to make (even educated) guesses.
- We have found  the ability to run continuous, fully automated test suites for more extensive evidence collection to be a major benefit of using CUTS.
- There has been significant interest shown by programs within Raytheon in the toolset since we have begun to socialize our work.

# Next Steps

- Enhanced CoWorkErs to include timing data in message payload to get actual deadline measurements rather than worst-case trends.
- A benchmark node agent to aggregate node test data and support remote test control (node, app failures etc.)
- Extend metrics collection to include additional items – application availability, middleware overhead, etc.

# Questions?