Continuous System Integration of Distributed Real-time and Embedded Systems

James H. Hill and Douglas Schmidt
Vanderbilt University
Nashville, TN, USA

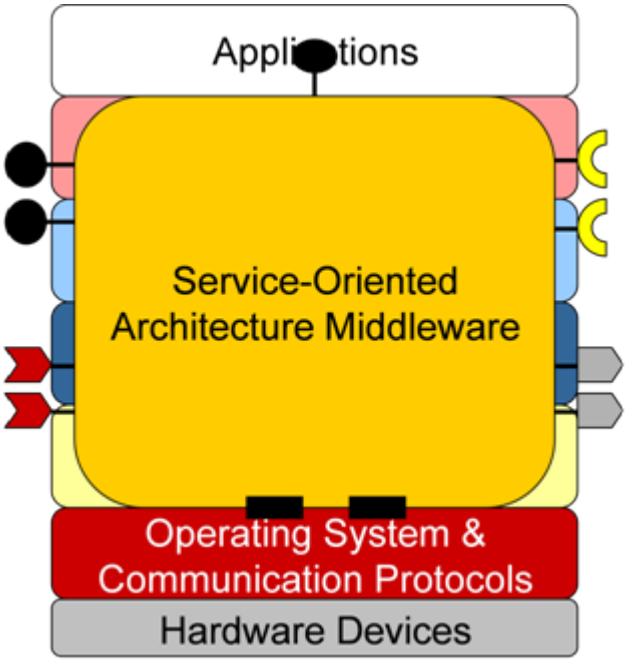
OMG's Workshop on Distributed Object Computing for Real-time and Embedded Systems (RTWS '08)
July 14 – 16, 2008
Washington, DC, USA

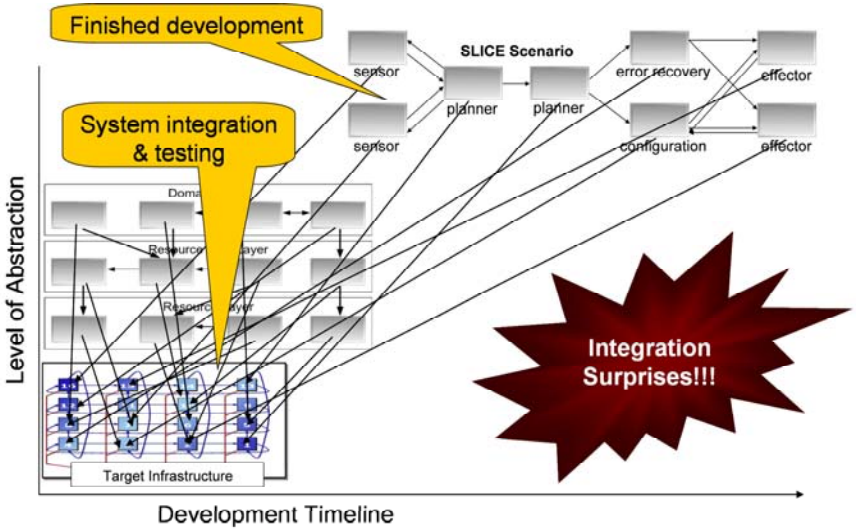
Institute for Software Integrated Systems

Vanderbilt University Nashville, Tennessee

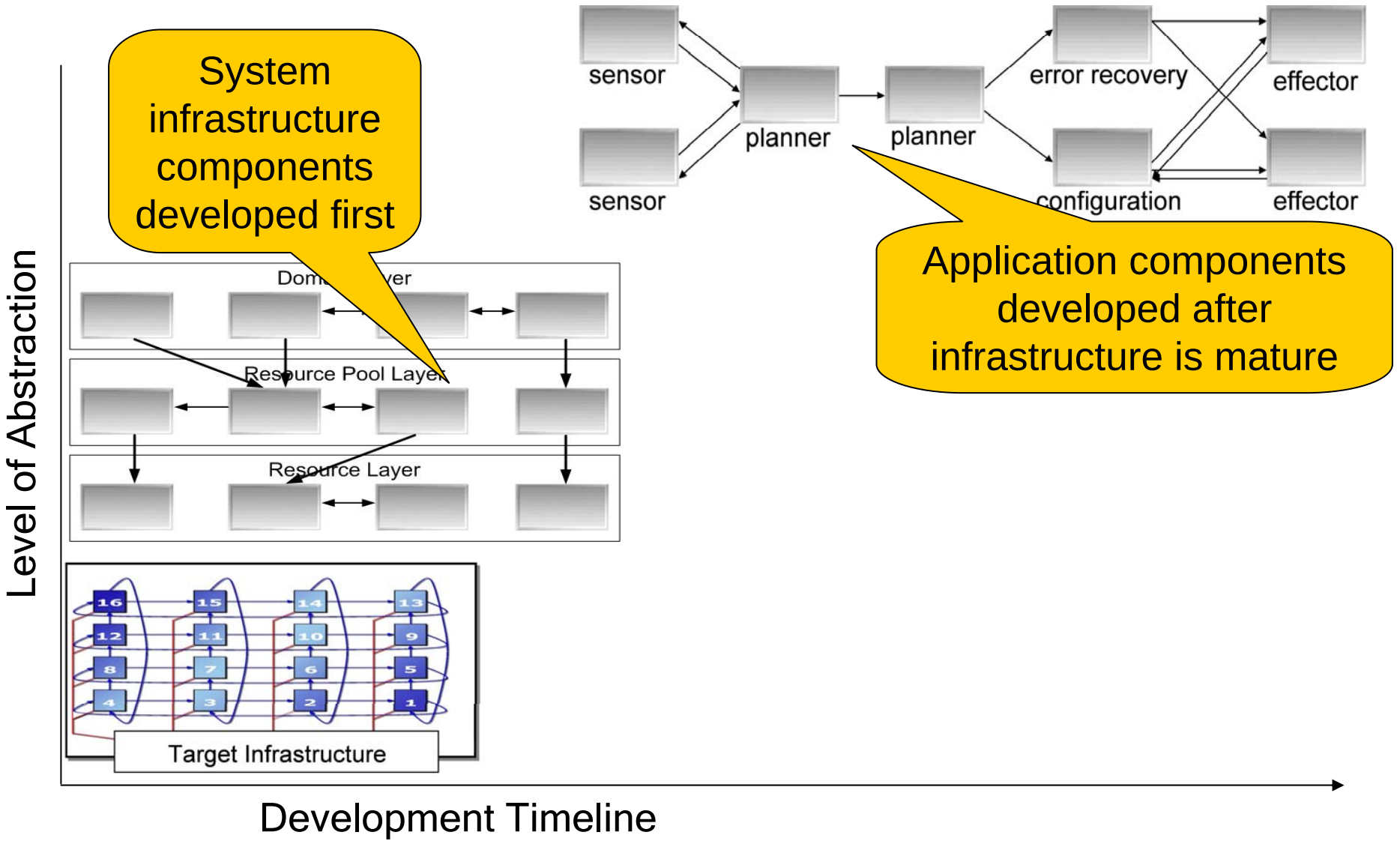
Motivation: SOA-based System Development

- Enterprise DRE systems are increasingly being developed using service-oriented architectures (SOAs)
  - *e.g.*, CCM, J2EE, & Microsoft .NET
- SOAs address many software development challenges
  - *e.g.*, reuse of core application-logic, improving application scheduability & reliability
- SOAs, however, incur unresolved problems that have adverse affects on development-time
  - *e.g.*, serialized-phased where application level components are not tested until long after infrastructure level components

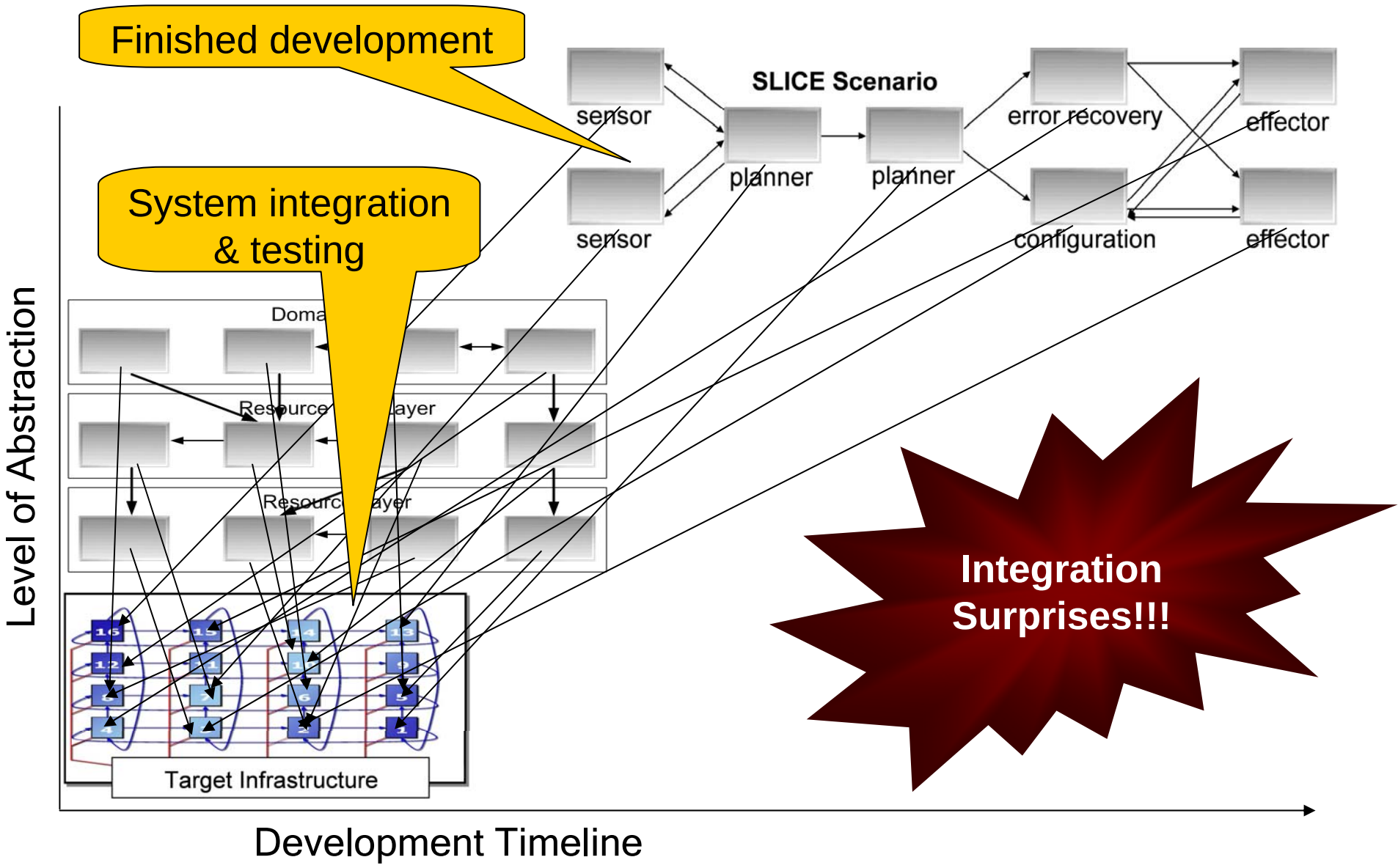




Serialized Phasing is Common in Large-scale Systems (1/2)



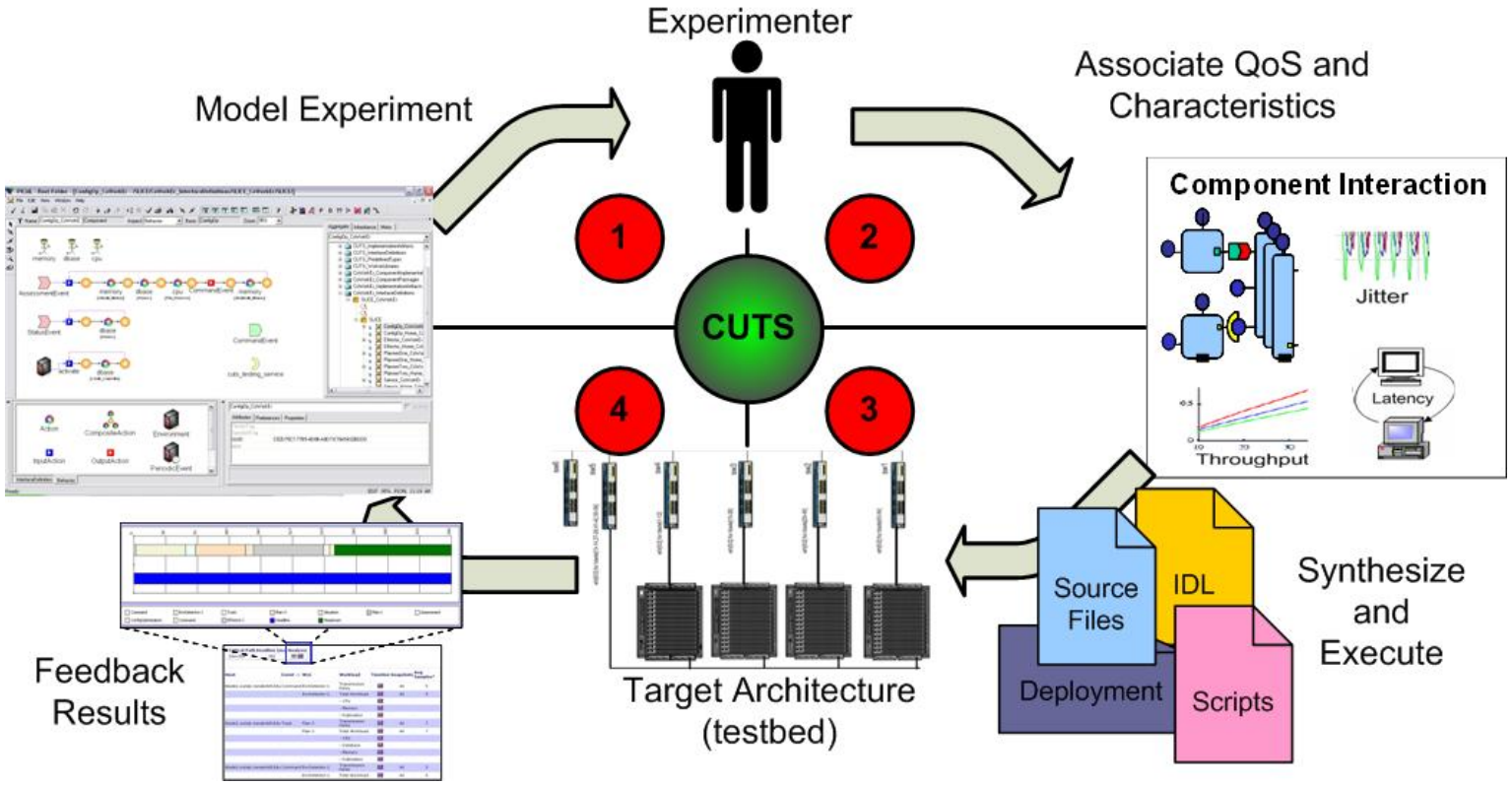
Serialized Phasing is Common in Large-scale Systems (2/2)



Model-Driven System Execution Modeling (SEM) Tools

- System execution modeling (SEM) tools are a promising technology for addressing serialized-phasing problems

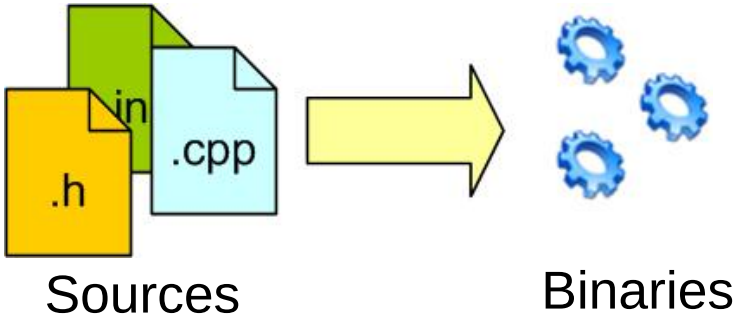
Component Workload Emulator (CoWorkEr) Utilization Test Suite (CUTS)
http://www.dre.vanderbilt.edu/CUTS



Enables early testing on target infrastructure throughout development lifecycle. SEM tools, however, have *limited testing capabilities* to support continuous system testing & evaluation.

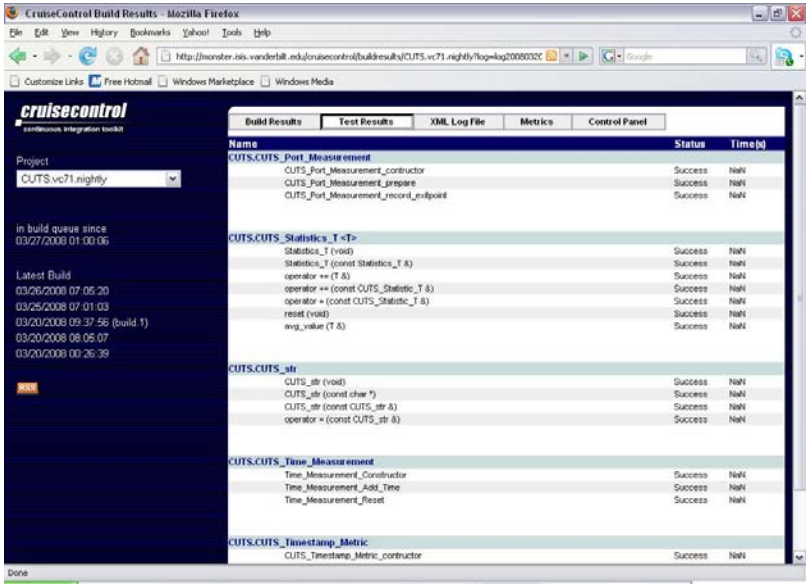
Solution Approach: Integrate SEM Tools with Continuous Integration Environments

- Continuous integration environments provide mechanisms that continuously validate software quality by:
  1. performing automated system builds upon source code commit or successful execution & evaluation of prior events



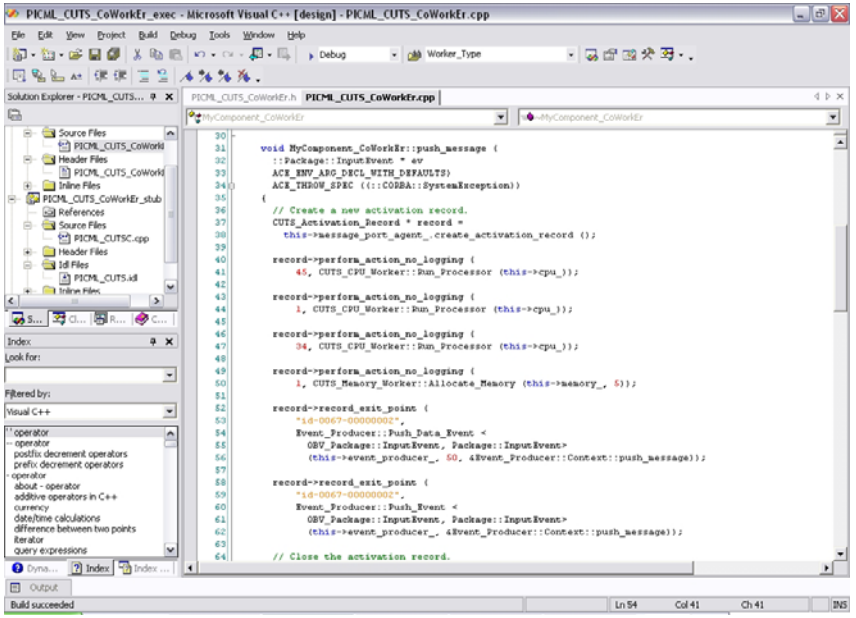
Solution Approach: Integrate SEM Tools with Continuous Integration Environments

- Continuous integration environments provide mechanisms that continuously validate software quality by:
  1. performing automated system builds upon source code commit or successful execution & evaluation of prior events,
  2. executing suites of unit tests to validate basic system functionality,



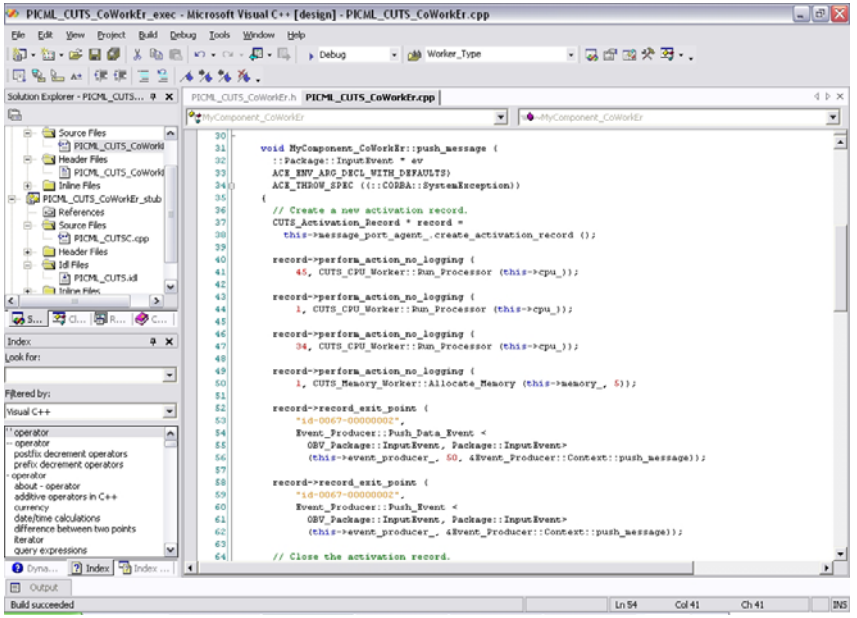
Solution Approach: Integrate SEM Tools with Continuous Integration Environments

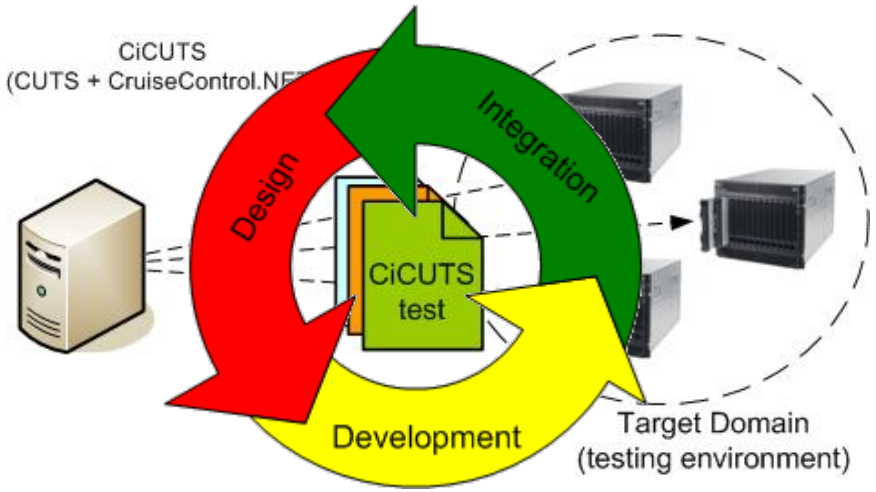
- Continuous integration environments provide mechanisms that continuously validate software quality by:
  1. performing automated system builds upon source code commit or successful execution & evaluation of prior events,
  2. executing suites of unit tests to validate basic system functionality,
  3. evaluating source code to ensure it meets coding standards, &
  4. executing code coverage analysis



Solution Approach: Integrate SEM Tools with Continuous Integration Environments

- Continuous integration environments provide mechanisms that continuously validate software quality by:
  1. performing automated builds commit or successful execution & evaluation of prior events,
  2. executing unit tests to validate basic system functionality,
  3. evaluating source code to ensure it meets coding standards, &
  4. executing code coverage analysis
- CiCUTS (*i.e.*, combination of continuous integration environments with CUTS) uses integration tests managed by continuous integration environments that dictate the behavior of CUTS

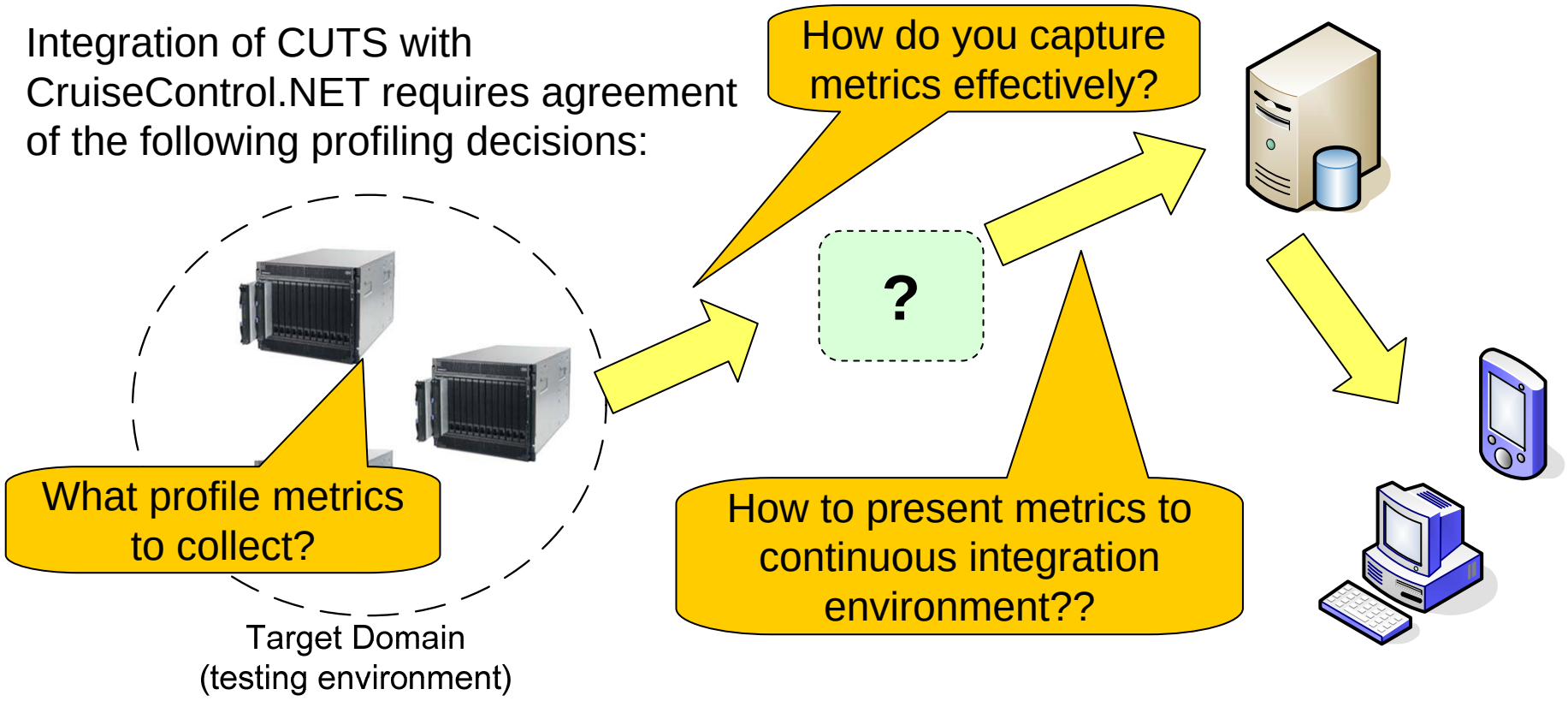




CiCUTS helps developers & testers ensure system QoS meets—or is close to meeting—its specification throughout the development lifecycle.

CiCUTS Integration Challenges

Integration of CUTS with CruiseControl.NET requires agreement of the following profiling decisions:



Integration Alternatives

- Extend profiling infrastructure of SEM tools to capture domain-specific metrics
- Capture domain-specific performance metrics in format understood by continuous integration environments
- Capture domain-specific performance metrics in an intermediate format

Alternative 1: Extending Profiling Infrastructure

Context

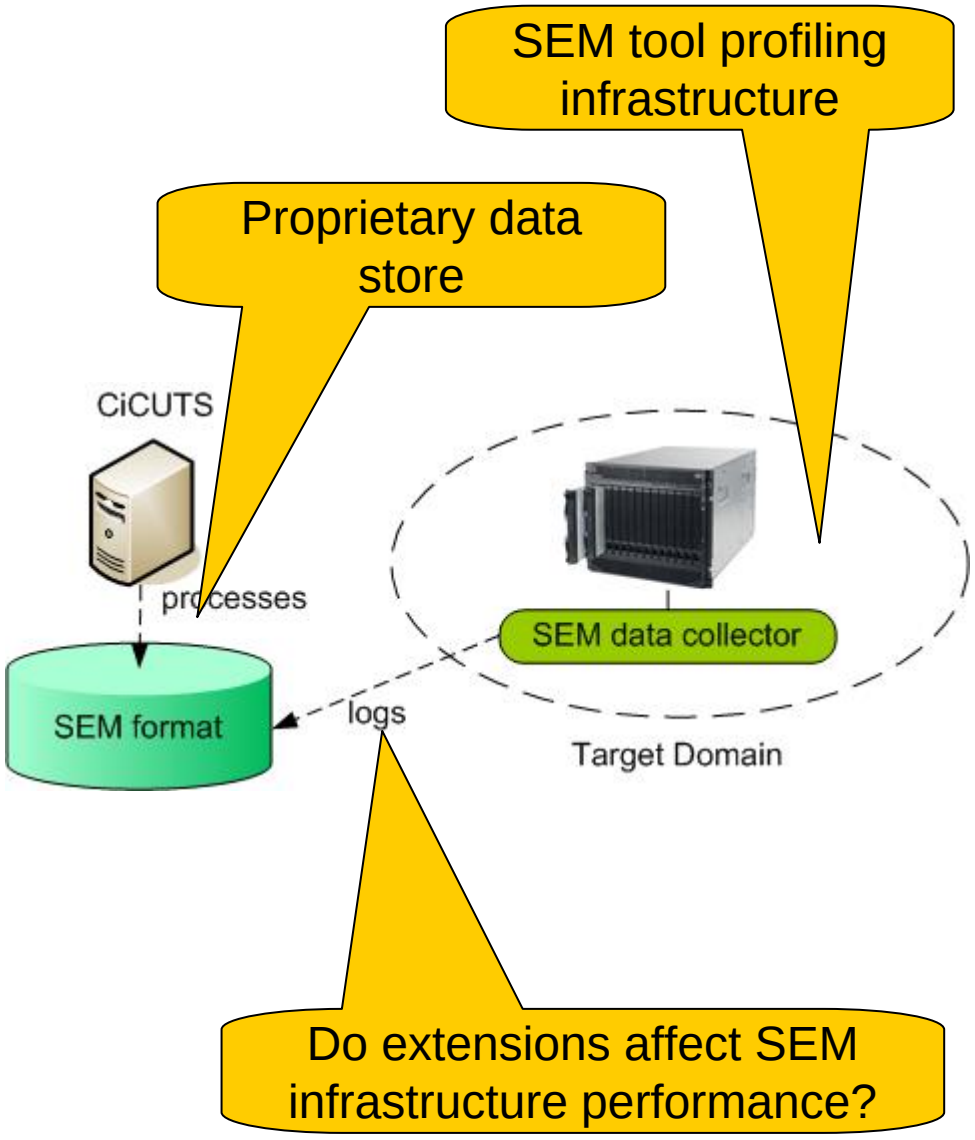
- SEM tools provide profiling infrastructures to collect predefined performance metrics
  - *e.g.*, execution times of events/function calls or values of method arguments

Advantages

- Simplifies development of profiling framework
  - *e.g.*, can leverage existing data collection techniques

Disadvantages

- Must ensure domain-specific metrics do not effect existing SEM tool performance
- SEM tools may be proprietary & extension may be prohibited



Alternative 2: Capture Metrics In Format Understood By Continuous Integration Environment

Context

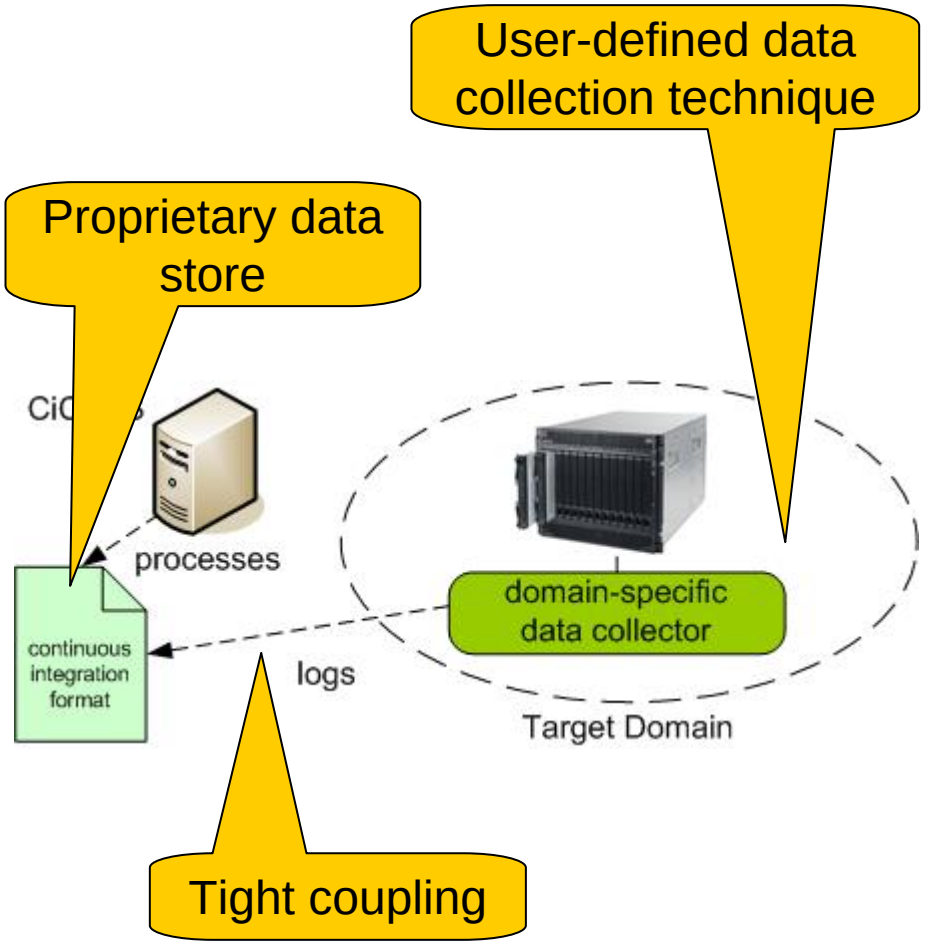
- Continuous integration environments use proprietary formats to store & analyze data
- May be feasible to collect & present metrics in format understood by continuous integration environments

Advantages

- Simplifies integration at the continuous integration side since format is known a priori

Disadvantages

- Requires a custom testing framework (adapter) to present data
- Tightly couples SEM tool with continuous integration environment



Alternative 3: Capture Metrics In Intermediate Format

Context

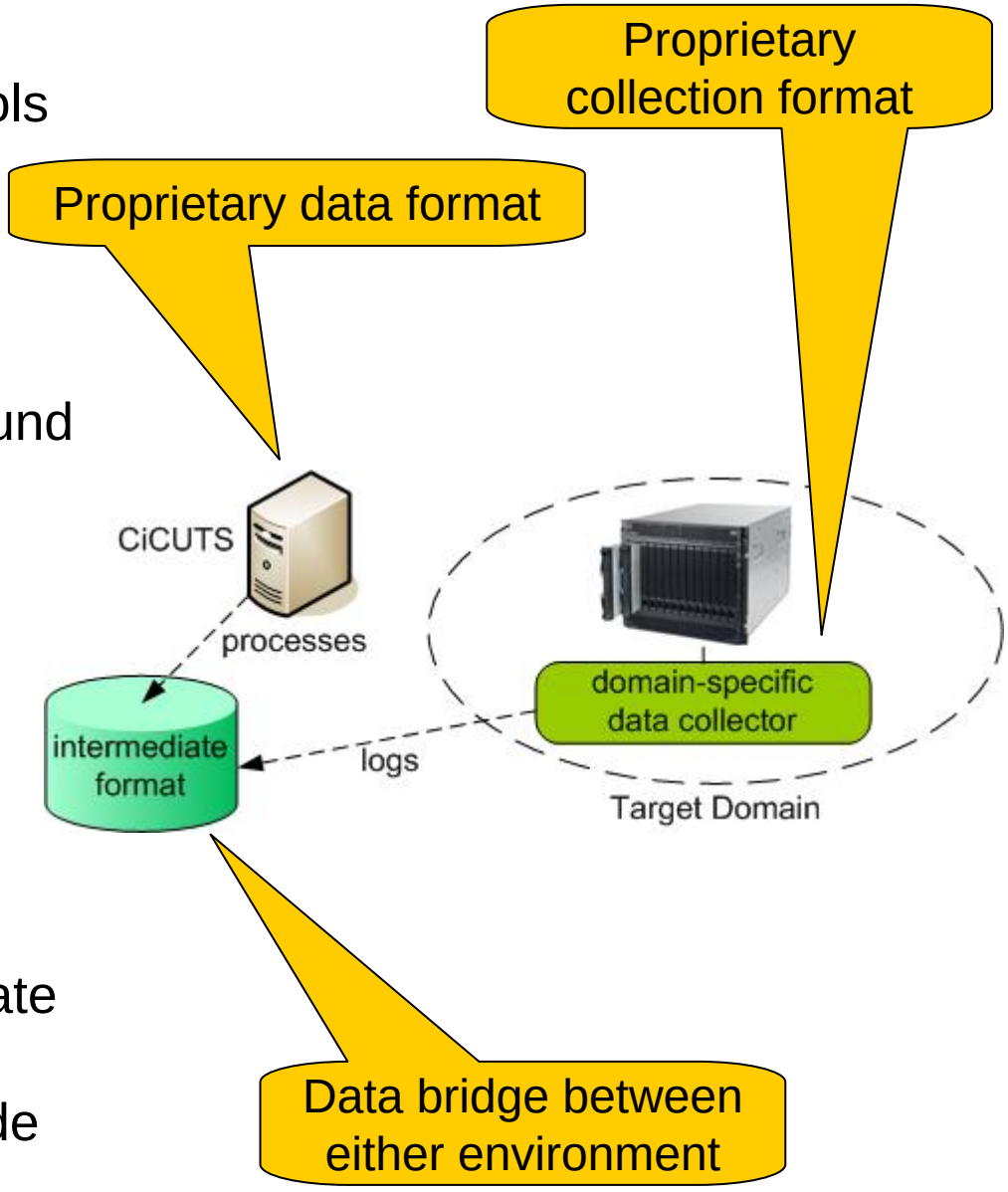
- Continuous integration & SEM tools each have proprietary methods
  - *e.g.*, data collection & representation
- May be feasible to store data in intermediate format that is not bound to a specific tool

Advantages

- Decouples continuous integration environment from the SEM tool
- Collection can be transparent to existing SEM tool infrastructure

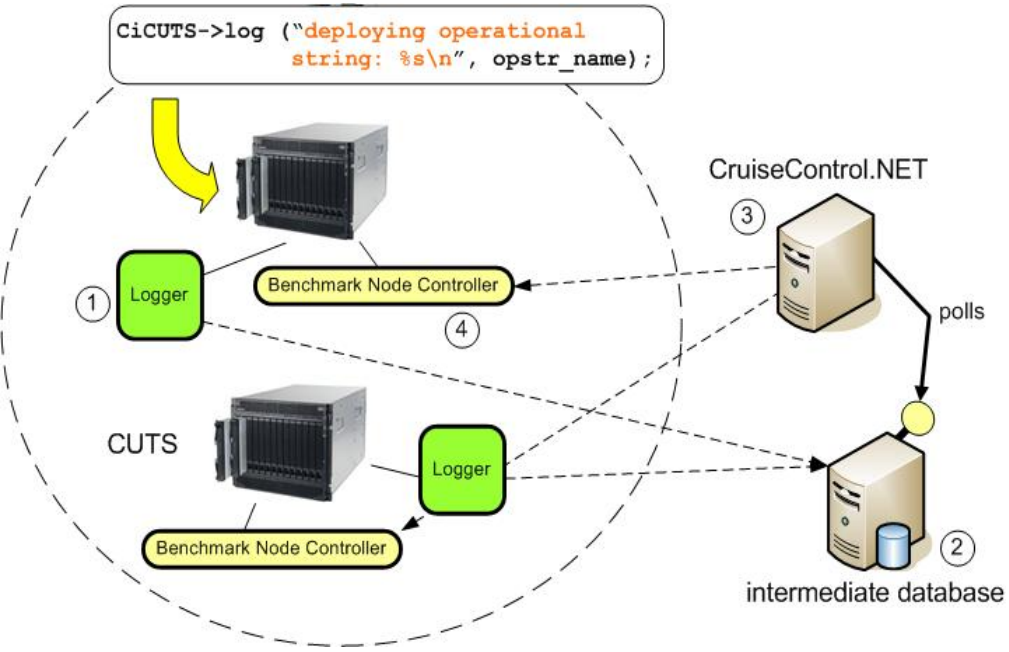
Disadvantages

- Requires agreement of intermediate format & implementation of data collectors & adapters on either side of the data store



Functionality & Structure of CiCUTS

- Chose integration alternative 3 because of its decouple design feature
  - e.g., developers can select different integration systems or SEM technologies, but leverage same technique

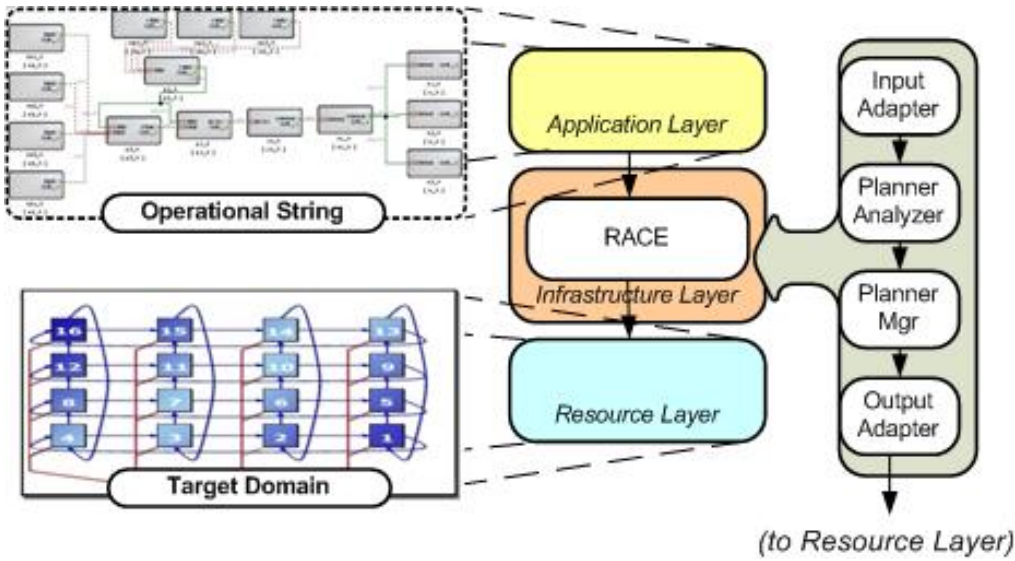


1. Loggers transparently capture domain-specific performance metrics via user-defined log messages
2. Intermediate database stores metrics collected by loggers for analysis
3. CruiseControl.NET executes & analyzes CUTS tests
4. Benchmark Node Controller execute commands received from CruiseControl.NET on the testing environment
   - *e.g.*, terminate container applications

Application of CiCUTS to an Enterprise DRE System

- RACE is a component-based DRE system that manages operational strings
- RACE supports two types of operational string deployments
  - **Static** – deployments created offline where components are assigned to hosts
  - **Dynamic** – deployments created online, but component assignment to host is based on operating conditions

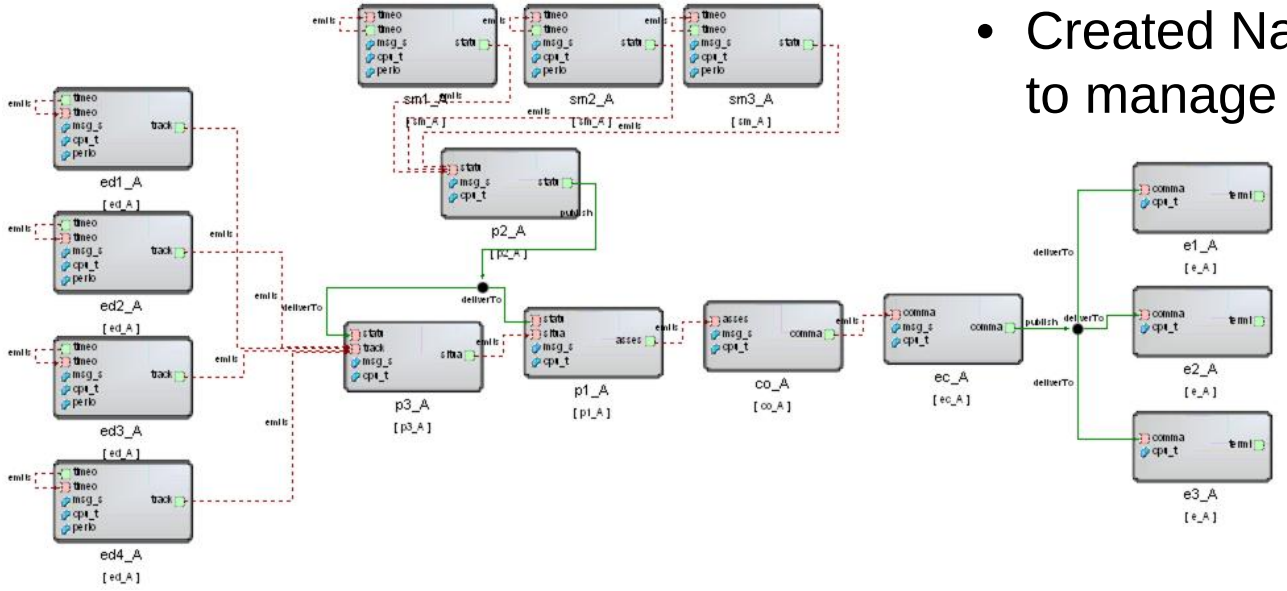
- *Baseline scenario* - higher priority operational strings must have longer lifetime than lower priority operational strings
  - e.g., under low resource availability



Experimental Design of RACE Baseline Scenario

CiCUTS Hypotheses

1. Developers can understand behavior & performance of infrastructure-level applications before system integration
2. Developers can ensure QoS performance is within specifications throughout the development lifecycle more efficiently & effectively than waiting until system integration to evaluate QoS performance



Experiment Design

- Constructed 10 identical operational strings with different importance values & used CUTS to generate implementation
  - A – H: 90
  - I – J: 2
- Augmented RACE source code with log messages for CiCUTS to intercept
- Created Nant scripts for CiCUTS to manage & execute
- *e.g.*, deploy/teardown operational string, send commands to Benchmark Node Controller, & query database for results

Hypothesis 1: Understanding Behavior & Performance of Infrastructure Level Components (1/2)

Metric Analysis Comparing RACE test 3285 with baseline test 3284

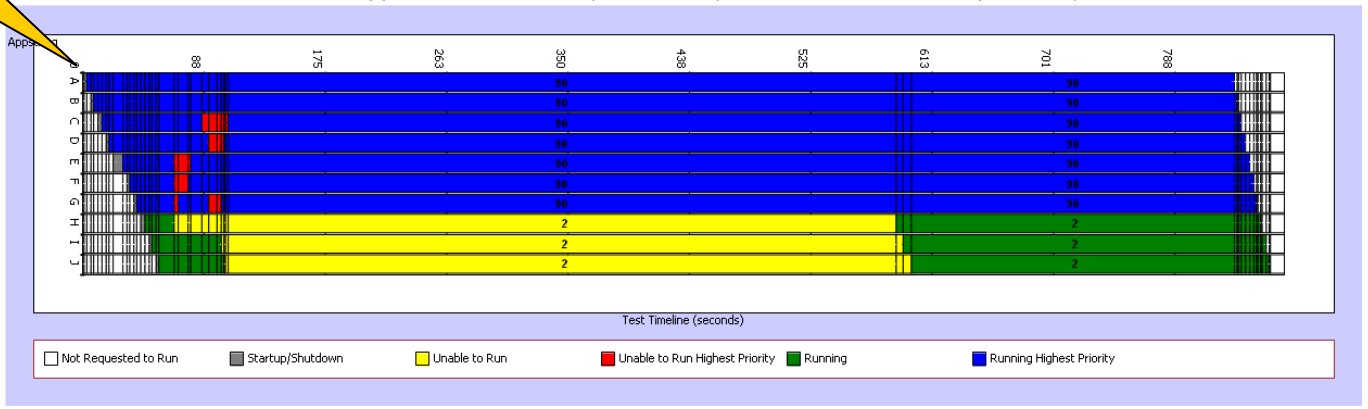
Return

	M1	M2
Race	93.743 %	625.227
Baseline	5.575 %	62.204
Improvement	15.090	10.051

Start of test

Dynamic Deployment Log Message Reconstruction

RACE Application Availability Timeline (numbers in bars are importance)



Start of test

Static Deployment Log Message Reconstruction

Baseline Application Availability Timeline (numbers in bars are importance)

Hypothesis 1: Understanding Behavior & Performance of Infrastructure Level Components (1/2)

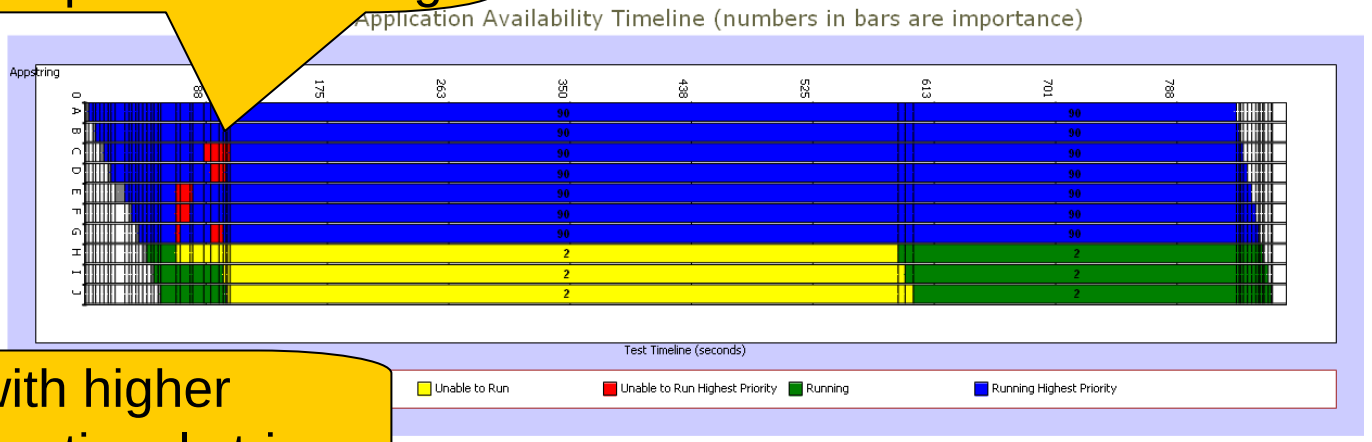
Metric Analysis Comparing RACE test 3285 with baseline test 3284

Return

	M1	M2
Race	93.743 %	625.227
Baseline	5.575 %	62.204
Improvement	15.090	10.051

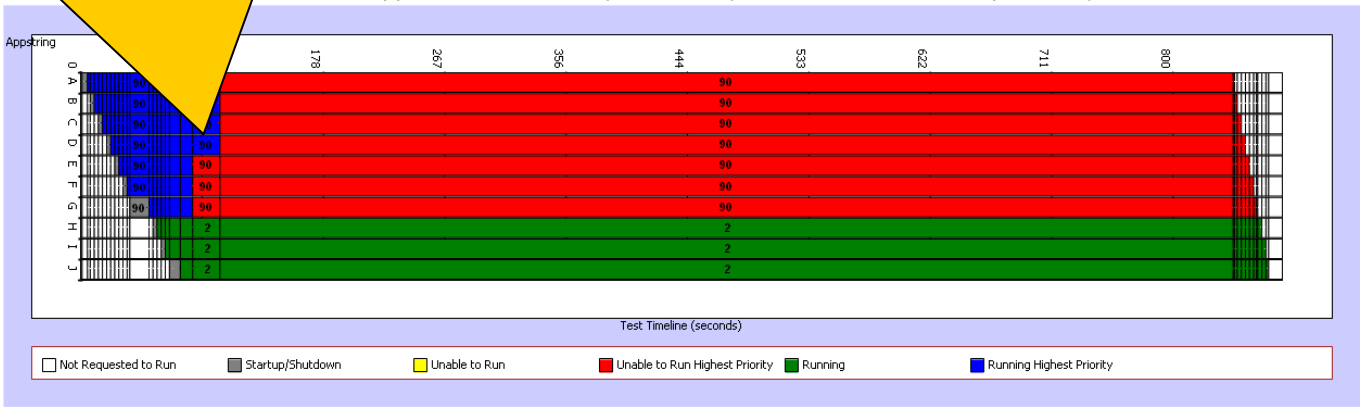
Kill node with higher importance operational string

Dynamic Deployment Log Message Reconstruction



Kill node with higher importance operational string

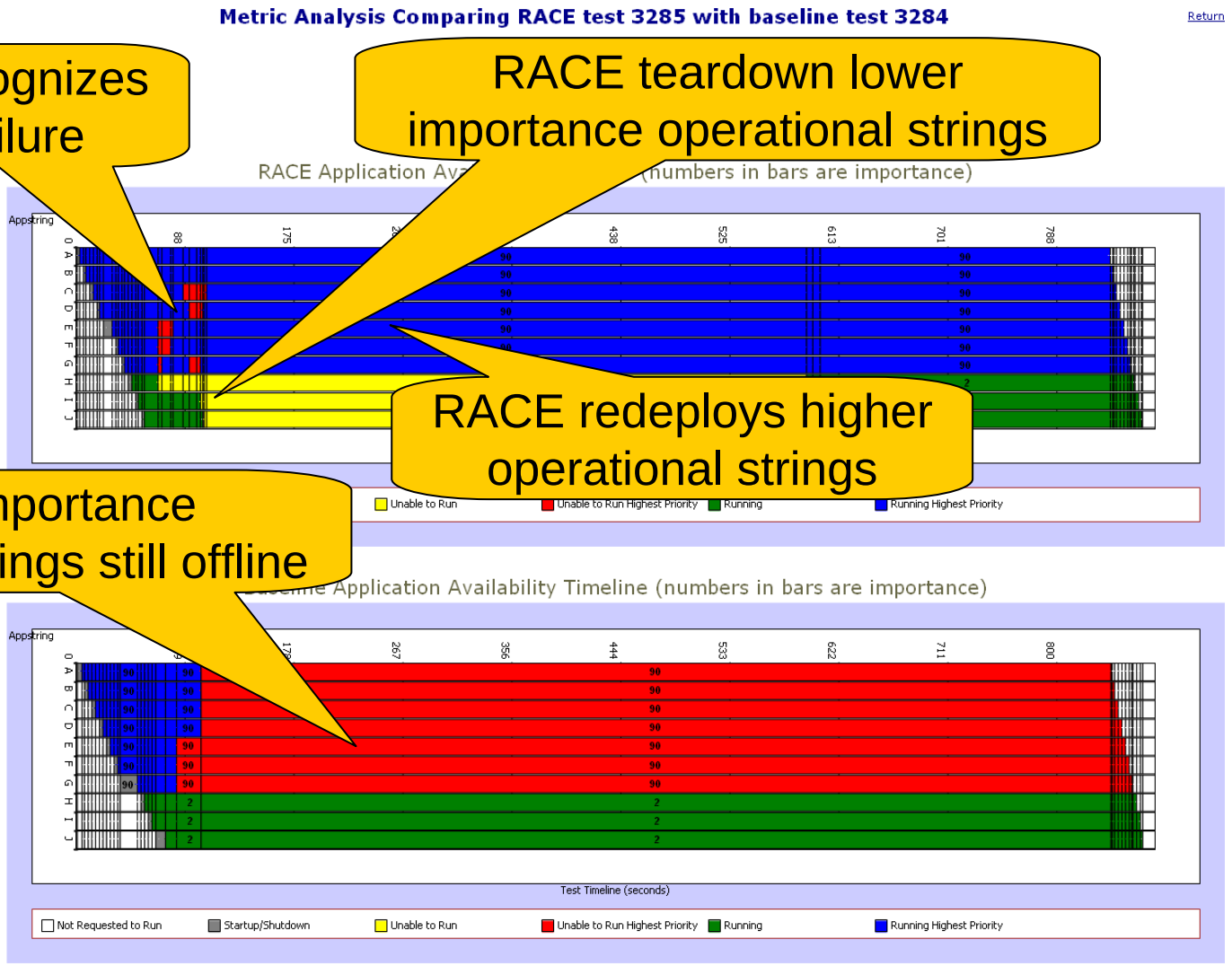
Static Deployment Log Message Reconstruction



Hypothesis 1: Understanding Behavior & Performance of Infrastructure Level Components (1/2)

Dynamic Deployment Log Message Reconstruction

Static Deployment Log Message Reconstruction



Hypothesis 1: Understanding Behavior & Performance of Infrastructure Level Components (1/2)

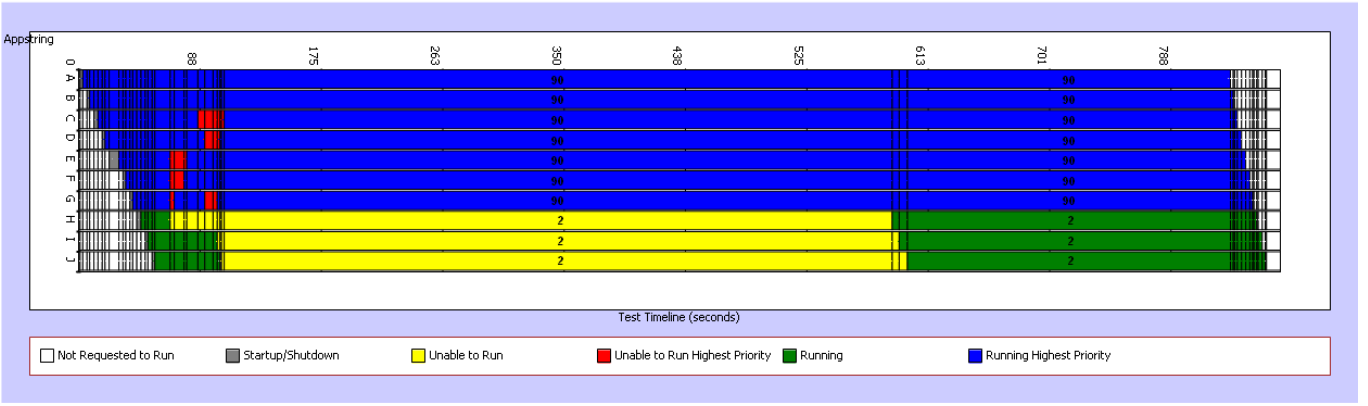
Dynamic Deployment Log Message Reconstruction

Metric Analysis Comparing RACE test 3285 with baseline test 3284

Return

	M1	M2
Race	93.743 %	625.227
Baseline	5.575 %	62.204
Improvement	15.090	10.051

RACE Application Availability Timeline (numbers in bars are importance)



Static Deployment Log Message Reconstruction

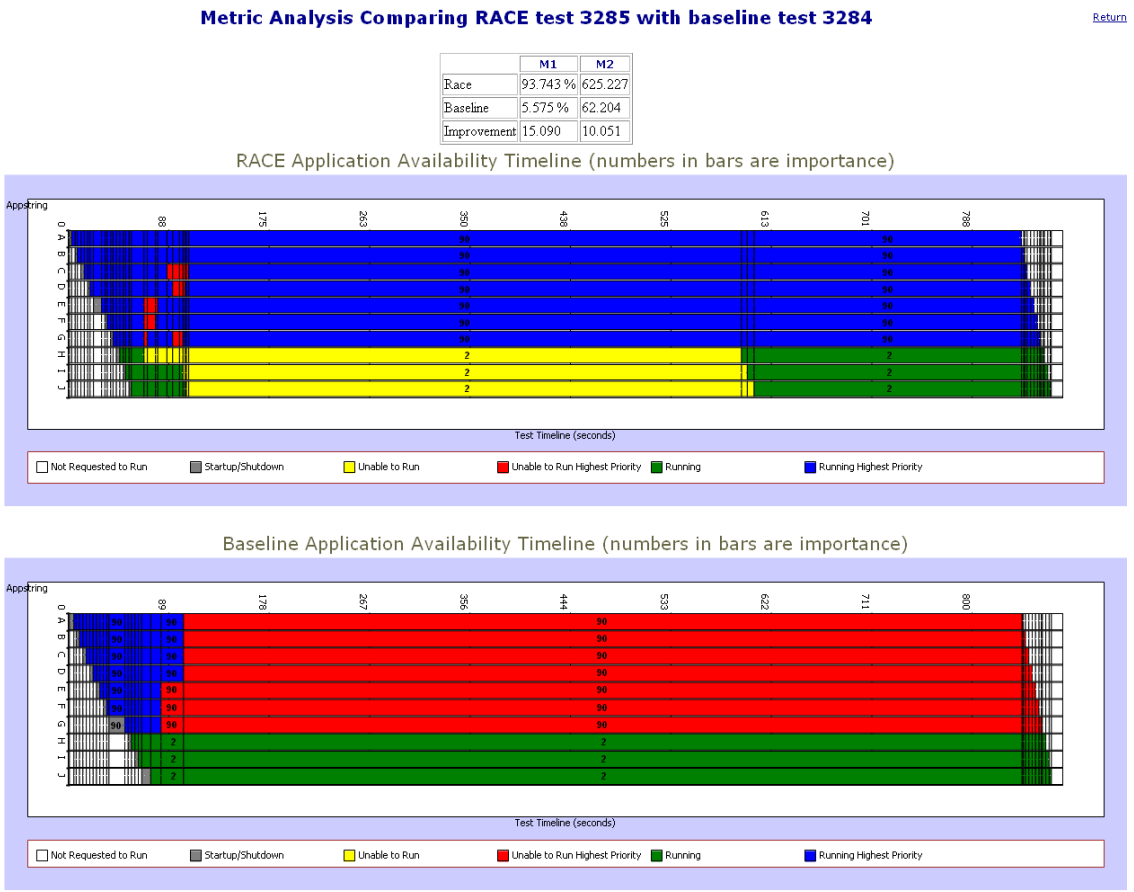
Baseline Application Availability Timeline (numbers in bars are importance)

The lifetime of higher importance operational strings is greater than the lifetime of lower importance operational strings

Hypothesis 1: Understanding Behavior & Performance of Infrastructure Level Components (2/2)

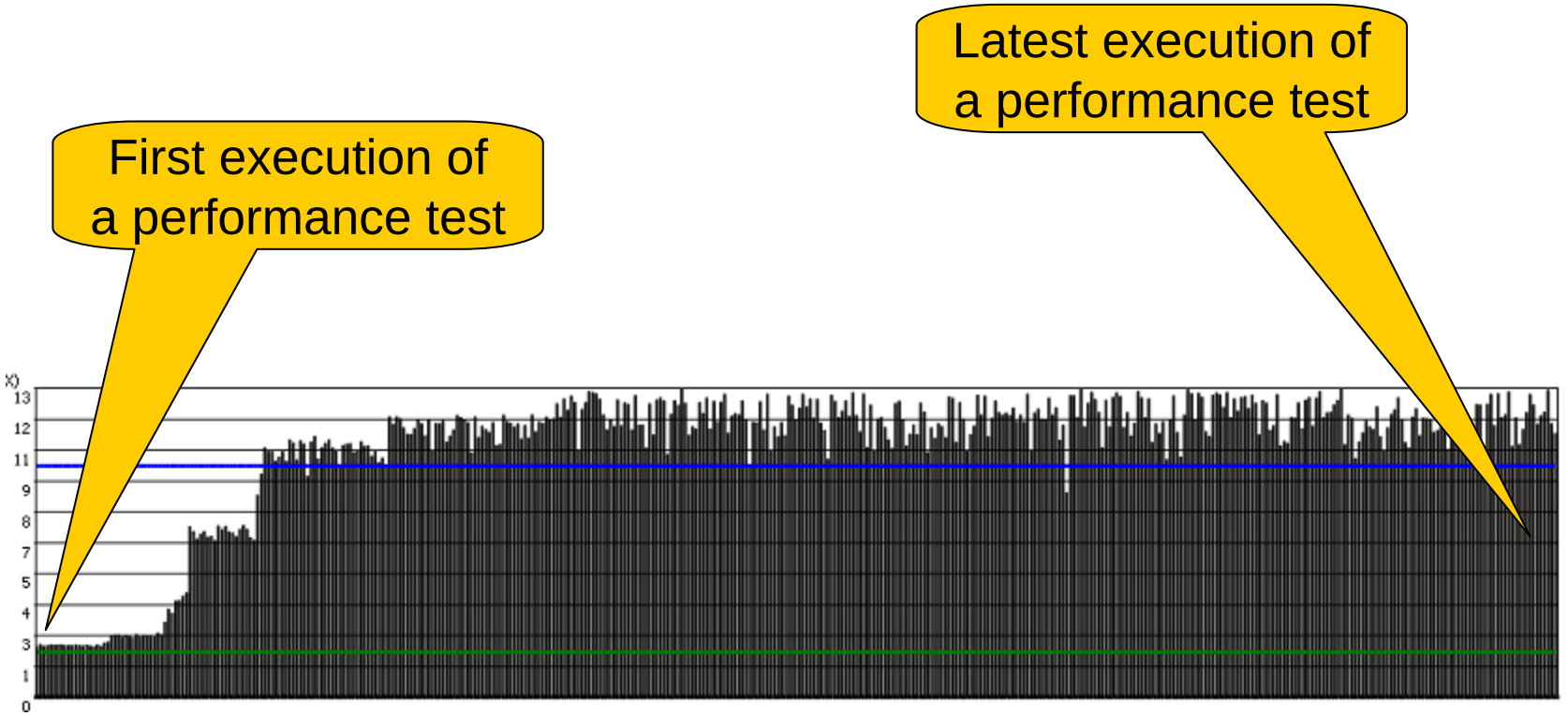
Benefits of CiCUTS

- Do not have to rely on *ad hoc* techniques to determine behavior of RACE
  - *e.g.*,manually inspecting & reconstructing distributed execution trace logs
- Simplified determining if RACE is performing as expected
- Performance evaluation of RACE can happen well before system integration time



Conclusion: CiCUTS helps developers understand the behavior & performance of infrastructure level components

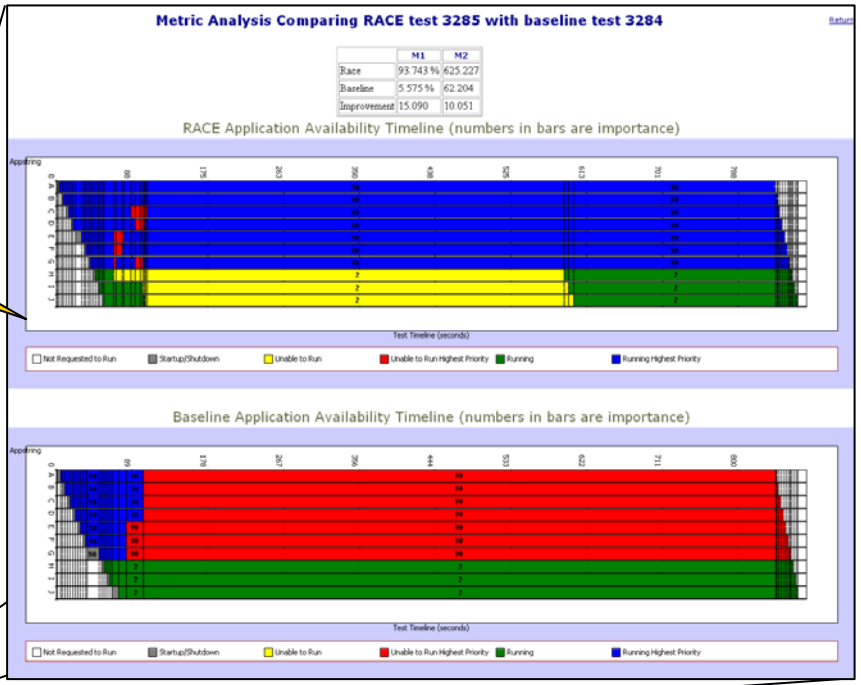
Hypothesis 2: Ensuring Infrastructure Performance is Within QoS Specifications

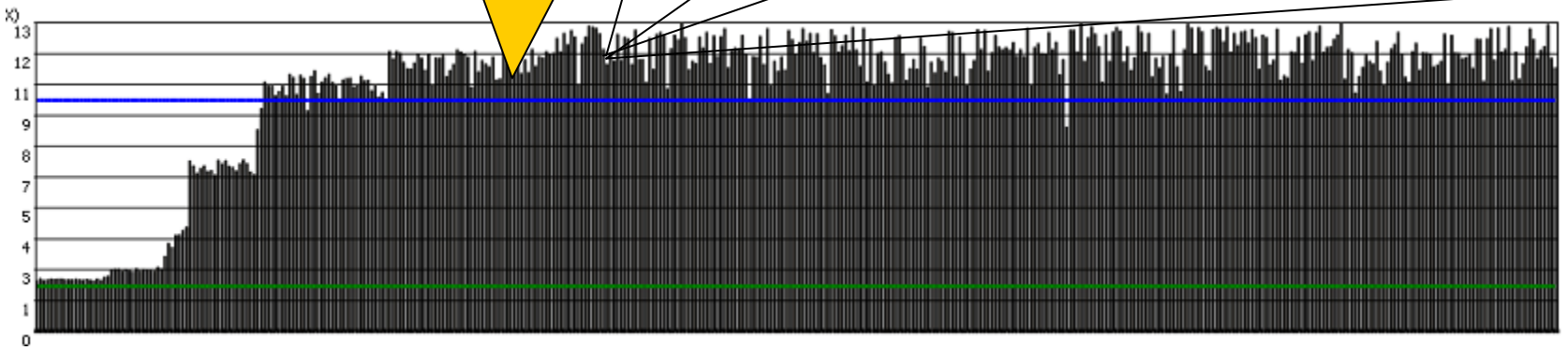


Hypothesis 2: Ensuring Infrastructure Performance is Within QoS Specifications

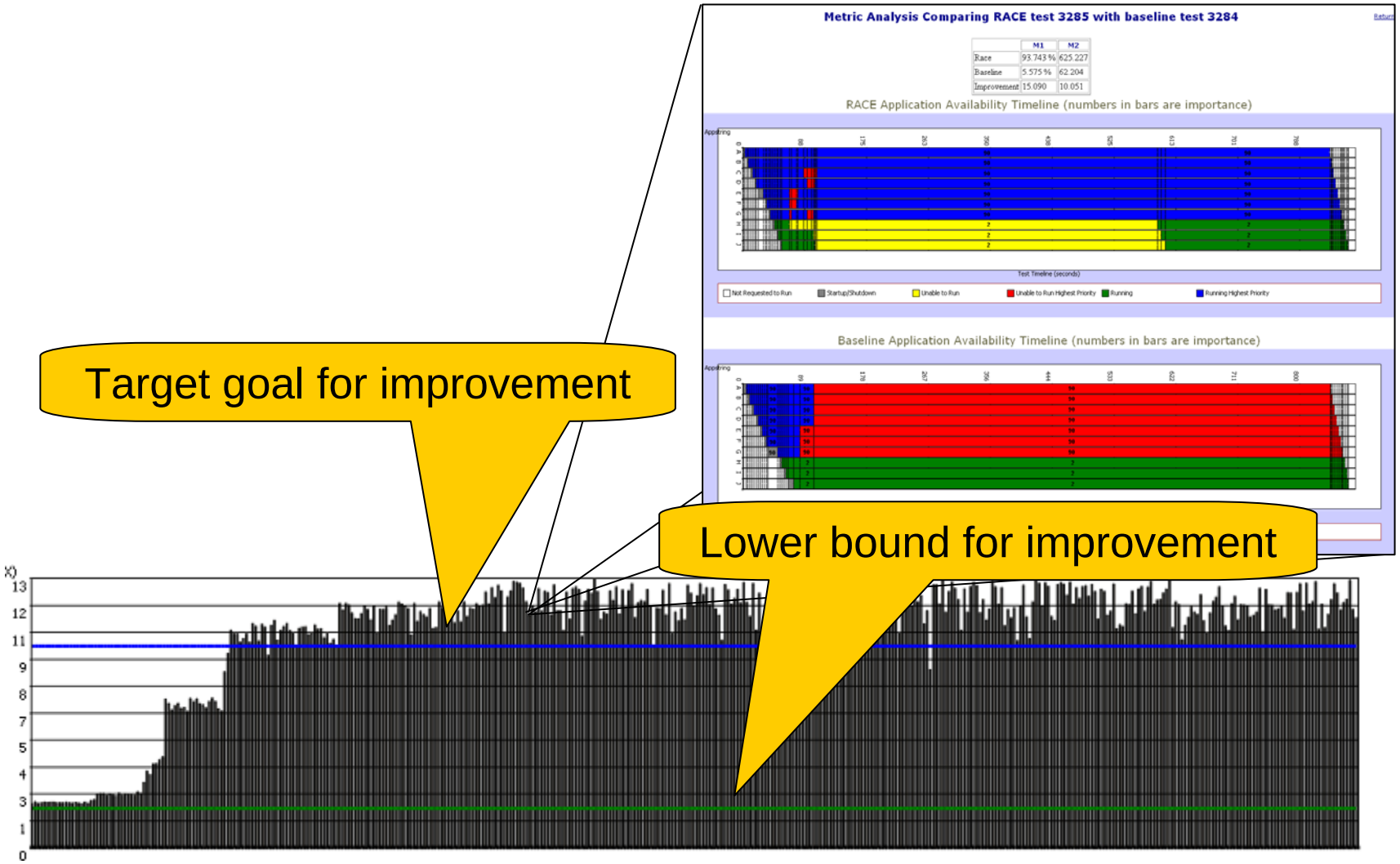
Single performance test of RACE

Bar height represents dynamically deployed operational string lifetime improvement





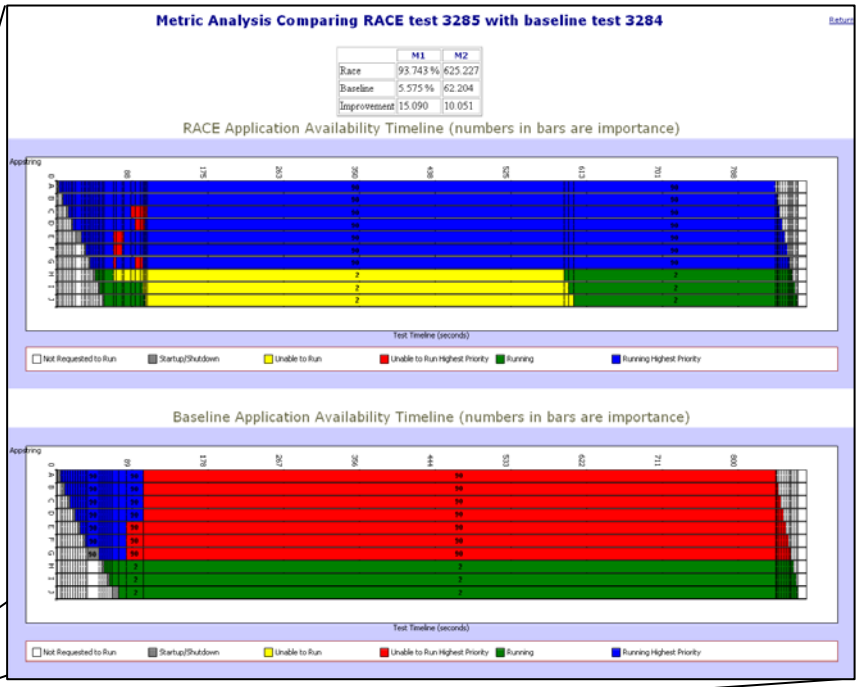
Hypothesis 2: Ensuring Infrastructure Performance is Within QoS Specifications

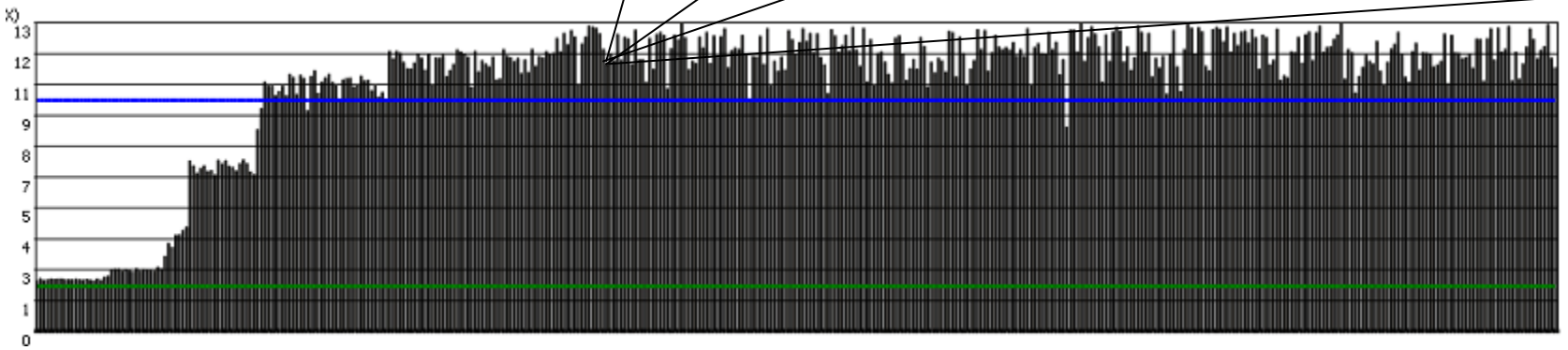


Hypothesis 2: Ensuring Infrastructure Performance is Within QoS Specifications

Benefits of CiCUTS

- Simplifies the process of continuous performance evaluation
- Does not require developers to:
  1. Monitor project's source for changes
  2. Update test environment
  3. Run performance tests
  4. Associate tests results with detected modifications



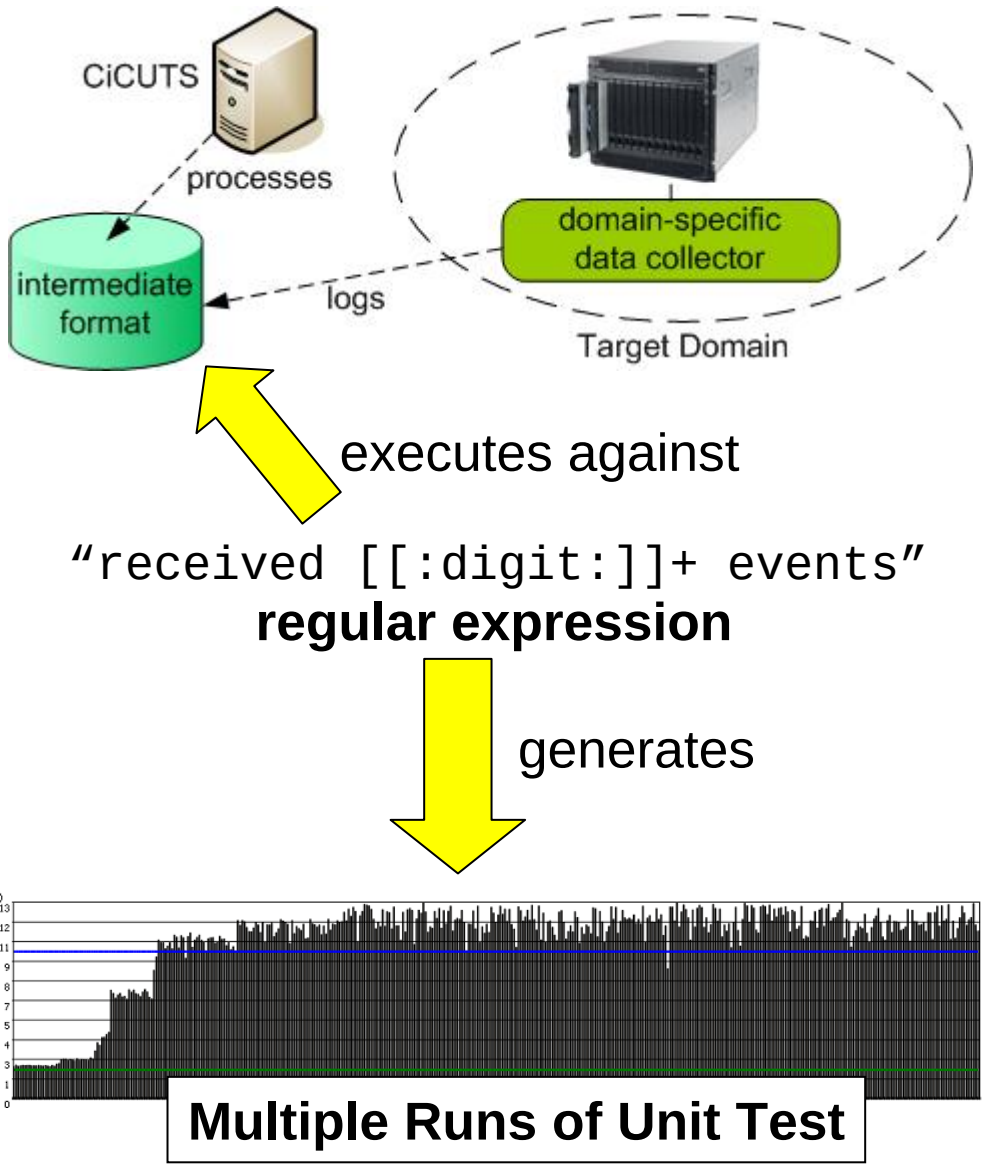


Conclusion: CiCUTS helps ensure infrastructure performance is within is QoS specficiations throughout the development lifecycle

Current & Future Research

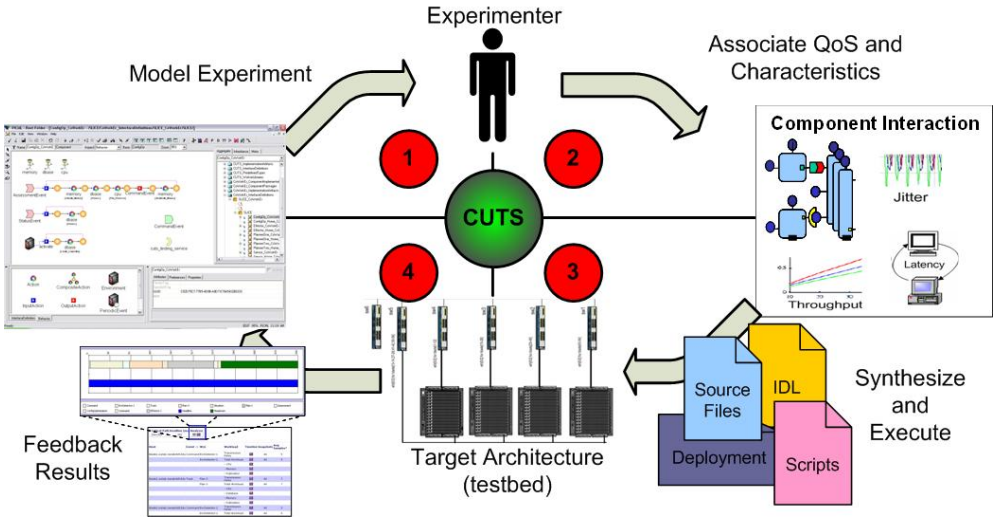
Generalization of Testing & Analysis Framework

- Messages used to construct behavior graphs can be identified using high-level constructs
  - e.g., “received {INT x} events”
- Given high-level constructs, data mining techniques can be applied to extract log messages of interest
- Values of interests can be extracted from log messages & used to generate behavior & performance graphs

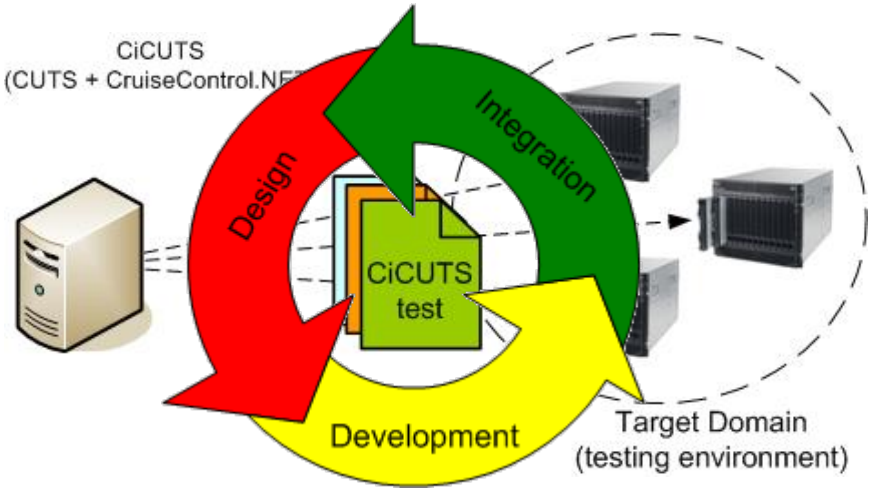


Concluding Remarks

- SEM tools provide mechanisms for executing performance tests during the early stages of development
- CiCUTS address the problem of improving testing capabilities for SEM tools via continuous integration systems



- CiCUTS is, therefore, able to help:
  1. Developers understand the behavior & performance of infrastructure level components
  2. Ensure infrastructure performance is within is QoS specficiations throughout the development lifecycle



CUTS & CiCUTS is available in open-source format at the following location
http://www.dre.vanderbilt.edu/CUTS

Questions