Model-Driven Engineering of Fault Tolerance Solutions in Enterprise Distributed Real-time and Embedded Systems

Sumant Tambe*

Aniruddha Gokhale

Jaiganesh Balasubramanian

Krishnakumar Balasubramanian

Douglas C. Schmidt

Vanderbilt University, Nashville, TN, USA

Contact : *sutambe@dre.vanderbilt.edu

Presented at OMG RTWS 2006

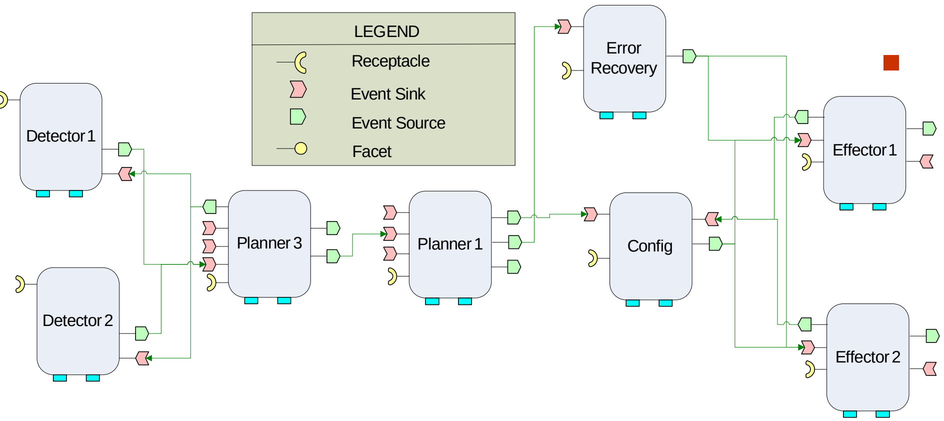
Arlington, VA, July 2006

This work is supported by subcontracts from LMCO & BBN

Vanderbilt University

Institute for Software Integrated Systems

Component-based Enterprise DRE Systems



- Characteristics of component-based enterprise DRE systems
  - Applications composed of one or more "*operational string*" of services
  - Composed of several services or systems of systems
  - Dynamic (re)-deployment of components into operational strings
  - Simultaneous QoS *(Reliability, Availability)* requirements
- Examples of Enterprise DRE systems
  - Advanced air-traffic control systems
  - Unmanned aerospace mission systems
  - Continuous patient monitoring systems

QoS Issues in Enterprise DRE Systems

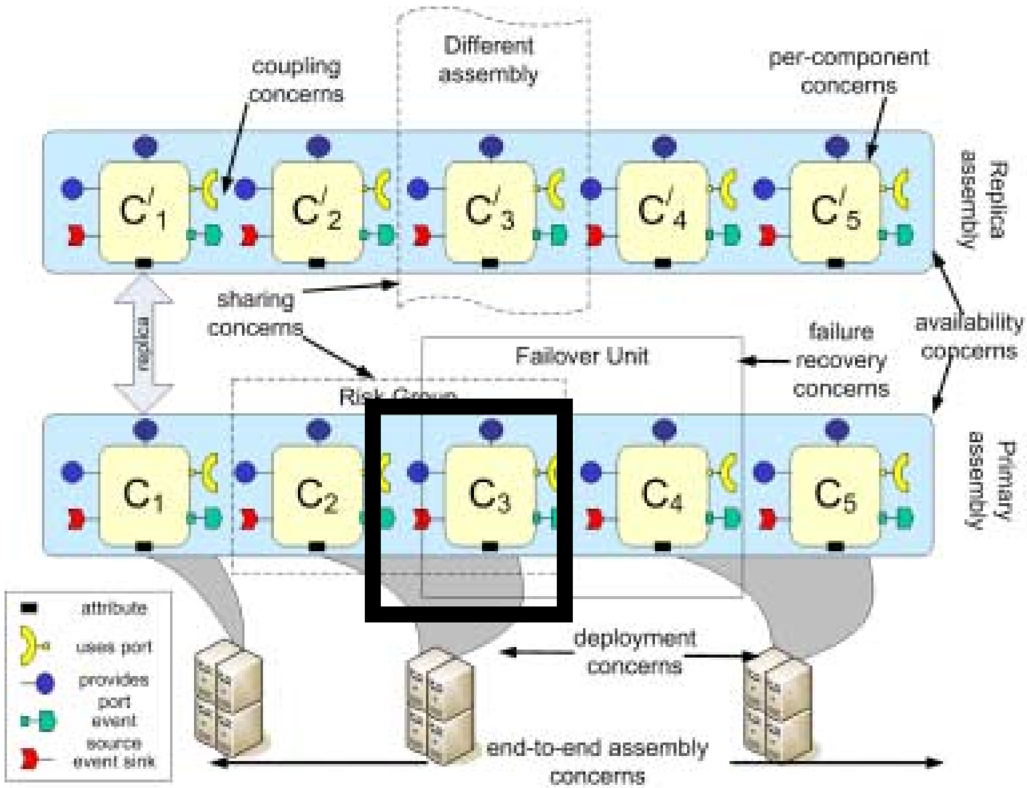
- Regulating & adapting to changes in runtime environments
- Satisfying tradeoffs between multiple (often tangled) QoS requirements
  - e.g., fault-tolerance, high performance, real-time etc.
- Satisfying QoS demands in the face of fluctuating and/or insufficient resources
  - e.g., hardware/software failures, mobile ad hoc networks

Changing Runtime

Tangled QoS Concerns

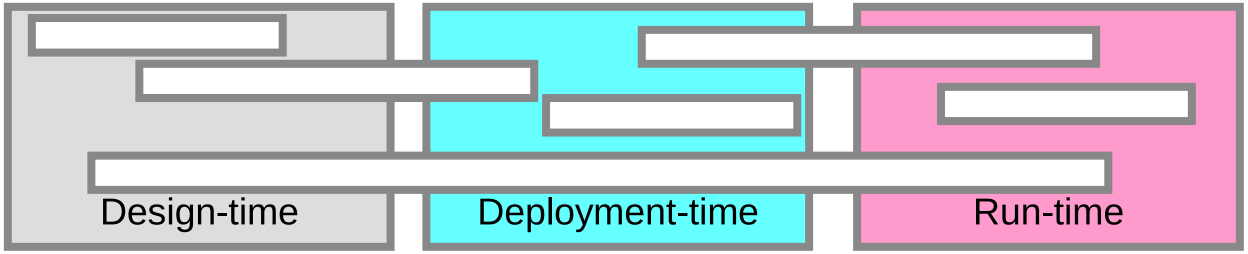
Resource Constrained

Tangled QoS-Concerns in Fault-Tolerance

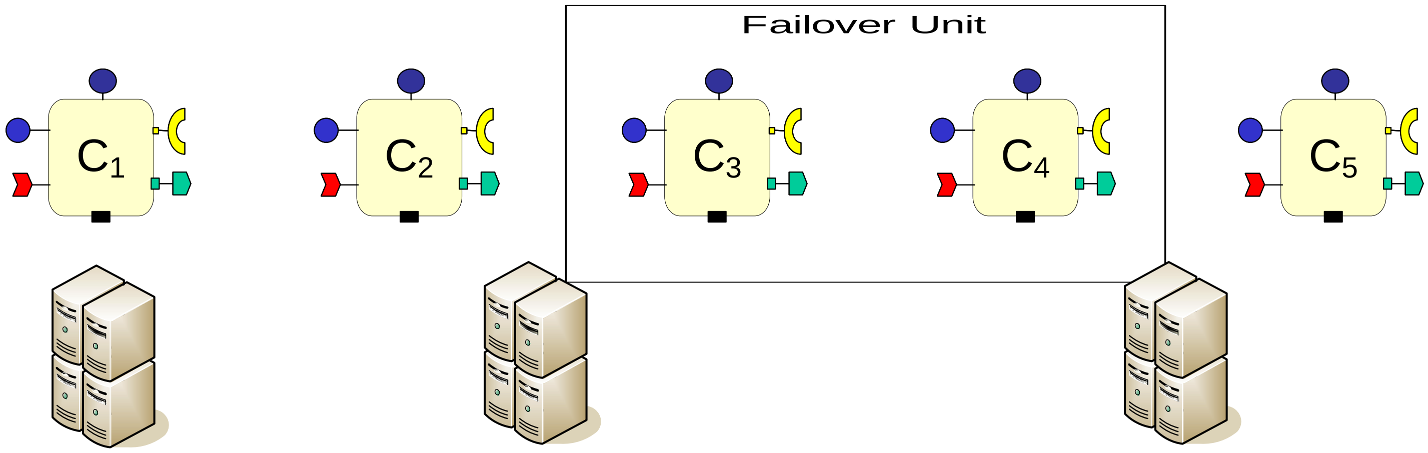


- Demonstrates numerous tangled para-functional concerns
- Significant sources of variability that affect end-to-end QoS (performance + availability)

Separation of Concerns & Managing Variability is the Key



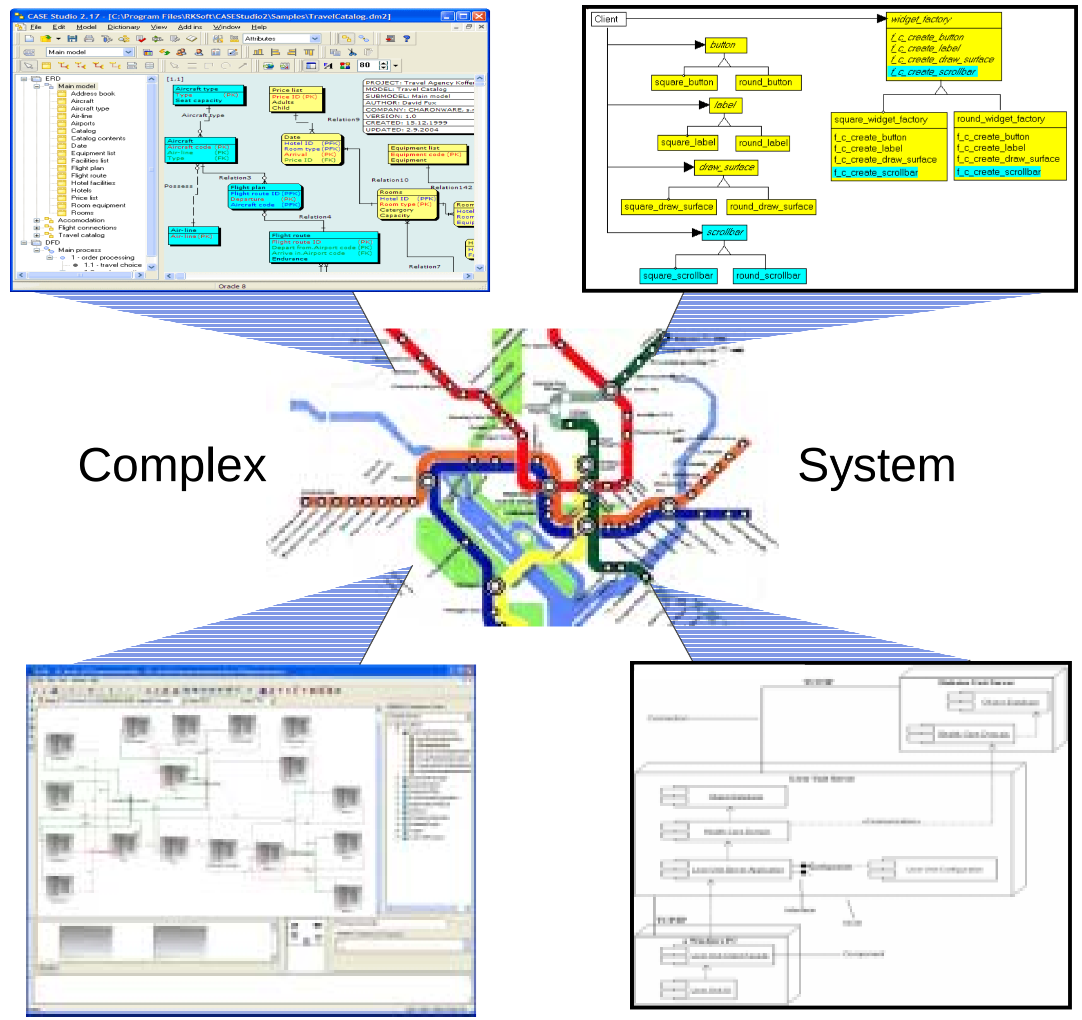
Fault-Tolerance Issues in DRE Systems



- **Per-component concern** – choice of implementation
  - Depends of resources, compatibility with other components in assembly
- **Availability concern** – what is the degree of redundancy? What replication styles to use? Does it apply to whole assembly?
- **Failure recovery concern** – what is the unit of failover?
- **State synchronization concerns –** What is data-sync frequency?
- **Deployment concern** – how to place components? Minimize failure risk to the system

Model-Driven Engineering – A Promising Approach

- Higher level of abstraction than third generation programming languages
- Model-per-concern paradigm alleviates system complexity
  - Deployment model
  - Component assembly model
  - System structural model
  - Different QoS models (Fault-tolerance)
- Generative and model transformation techniques to weave in appropriate glue code



Fault-tolerance Modeling Abstractions

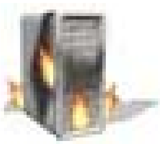
PICML (Platform Independent Component Modeling Language) Metamodel Enhancements made to CoSMIC Modeling Tool suite

- **Fail-over Unit (FOU):** Abstracts away details of granularity of protection (e.g. Component, Assembly, App-string)
- **Replica Group (RPG):** Abstracts away fault-tolerance policy details (e.g. Active/passive replication, state-synchronization, topology of replica)
- **Shared Risk Group (SRG):** Captures associations related to risk. (e.g. shared power supply among processors, shared LAN)

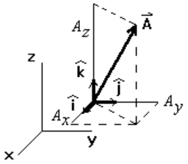
Interpreter (component placement constraint solver): Encapsulates an algorithm for component-node assignment based on replica distance metric

Protection granularity concerns

State-synchronization concerns

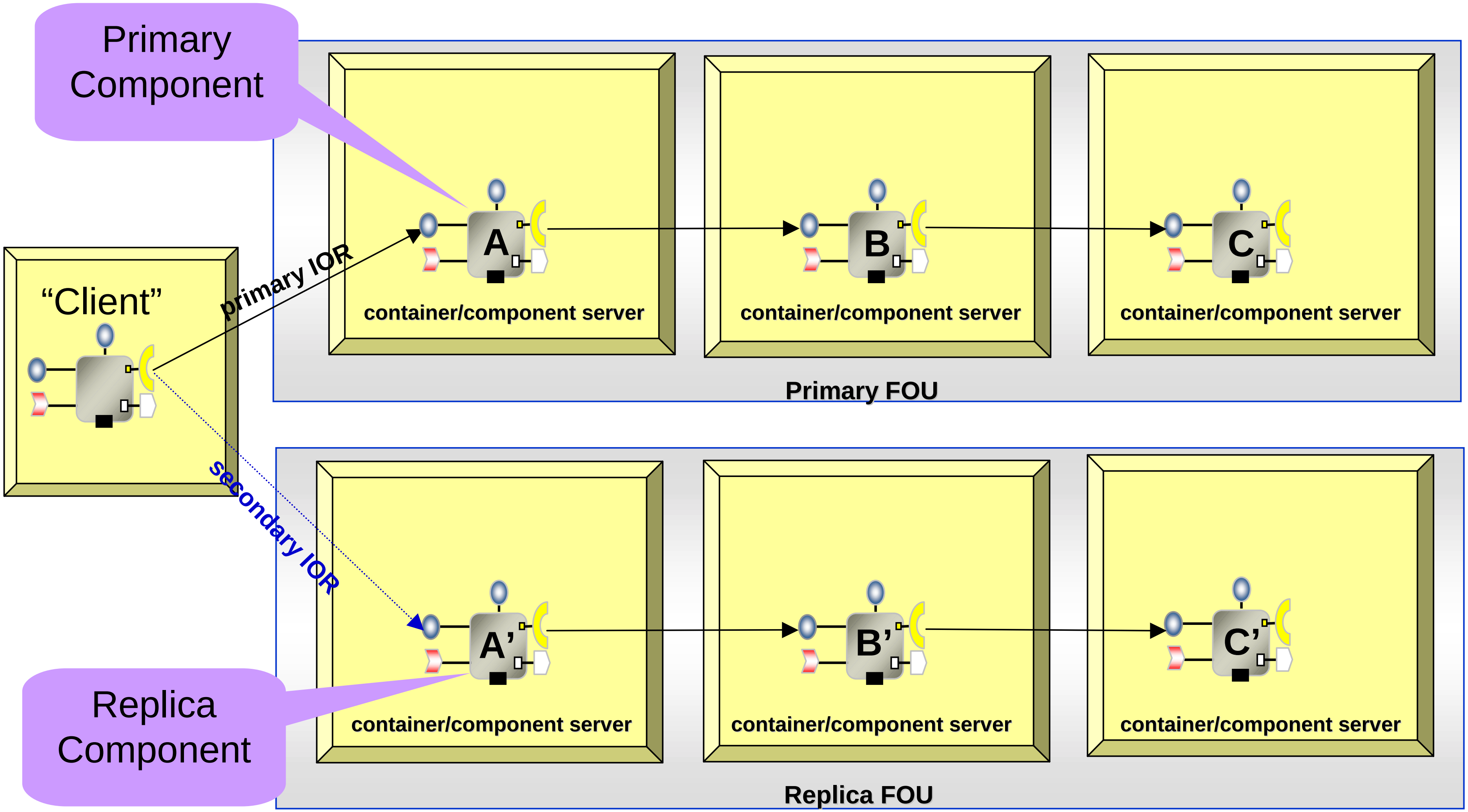


Component Placement constraints

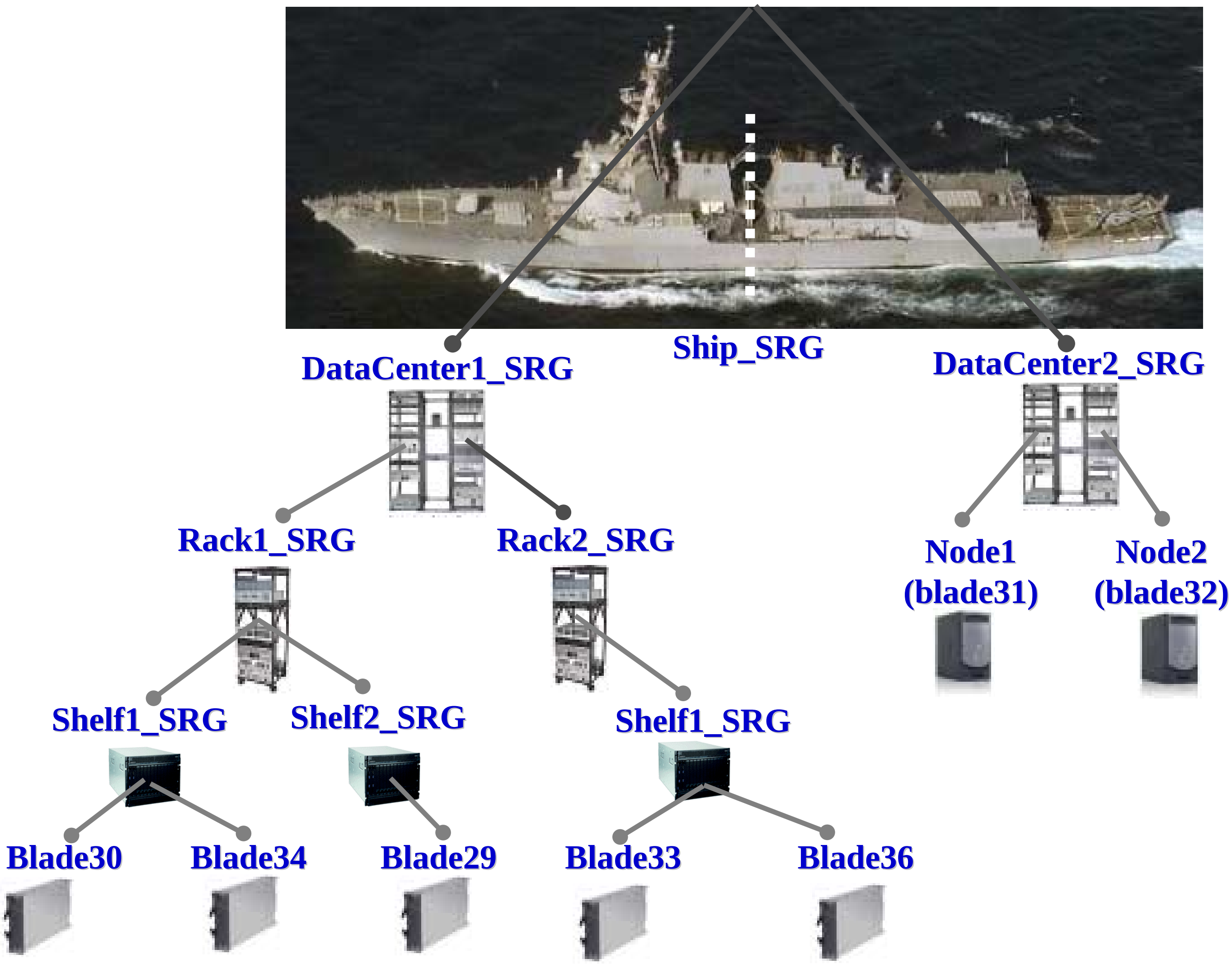


Replica Distance Metric

Fail-over Unit Example



Shared Risk Group Example

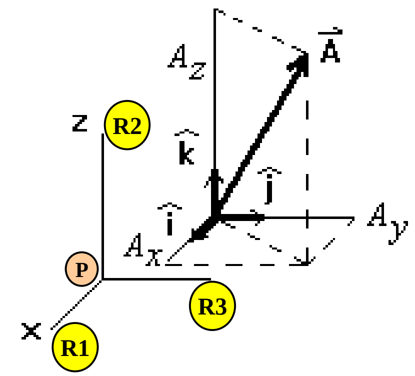


Formulation of Replica Placement Problem

Define N orthogonal vectors, one for each of the distance values computed for the N components (with respect to a primary) and vector-sum these to obtain a resultant. Compute the magnitude of the resultant as a representation of the composite distance captured by the placement

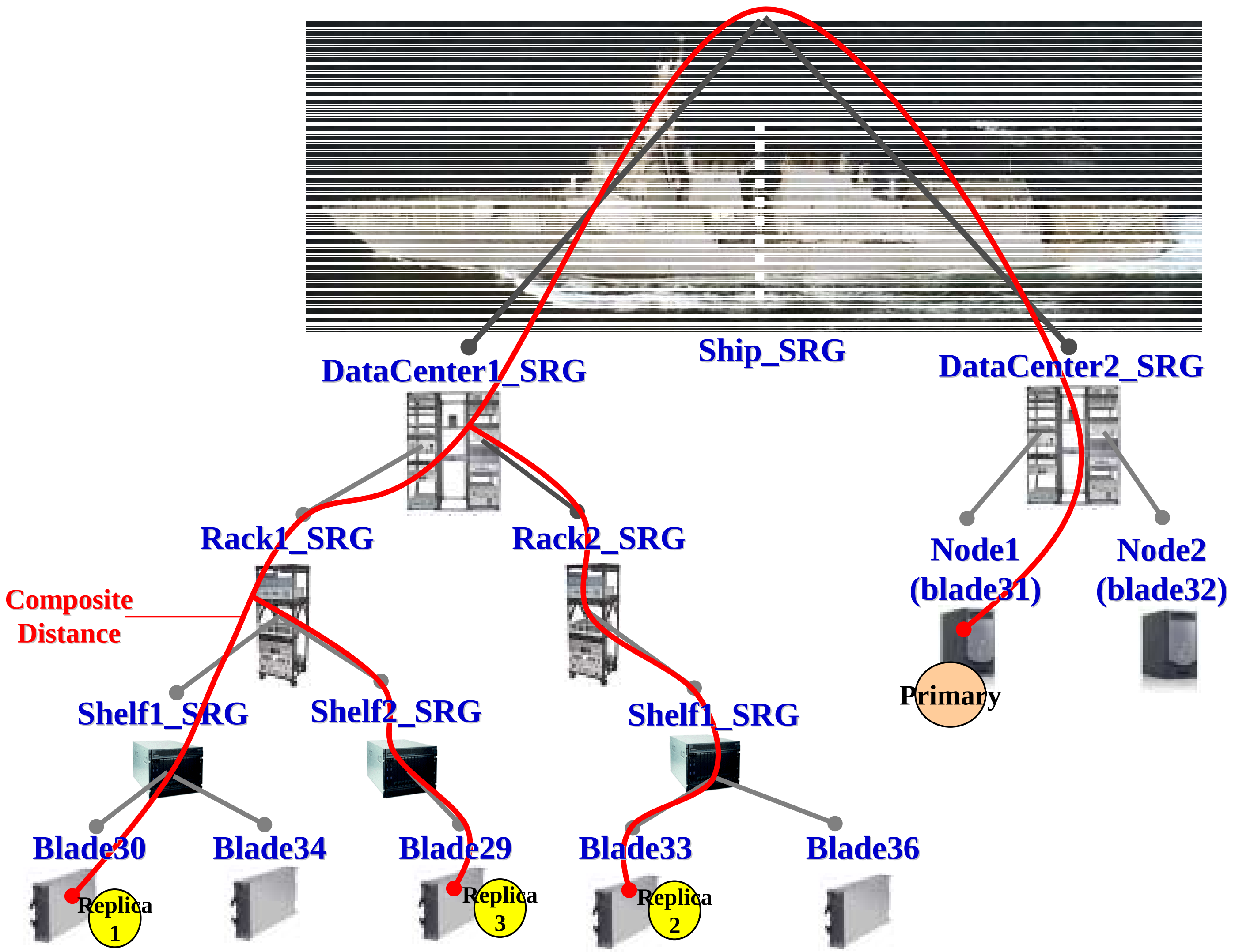
.

1. Compute the distance from each of the replicas to the primary for a placement.
2. Record ***each* distance as a vector, where** all vectors are orthogonal.
3. Add the vectors to obtain a resultant.
4. Compute the magnitude of the resultant.
5. Use the resultant in all comparisons (either among placements or against a threshold)
6. Apply a penalty function to the composite distance (e.g. pair wise replica distance or uniformity)



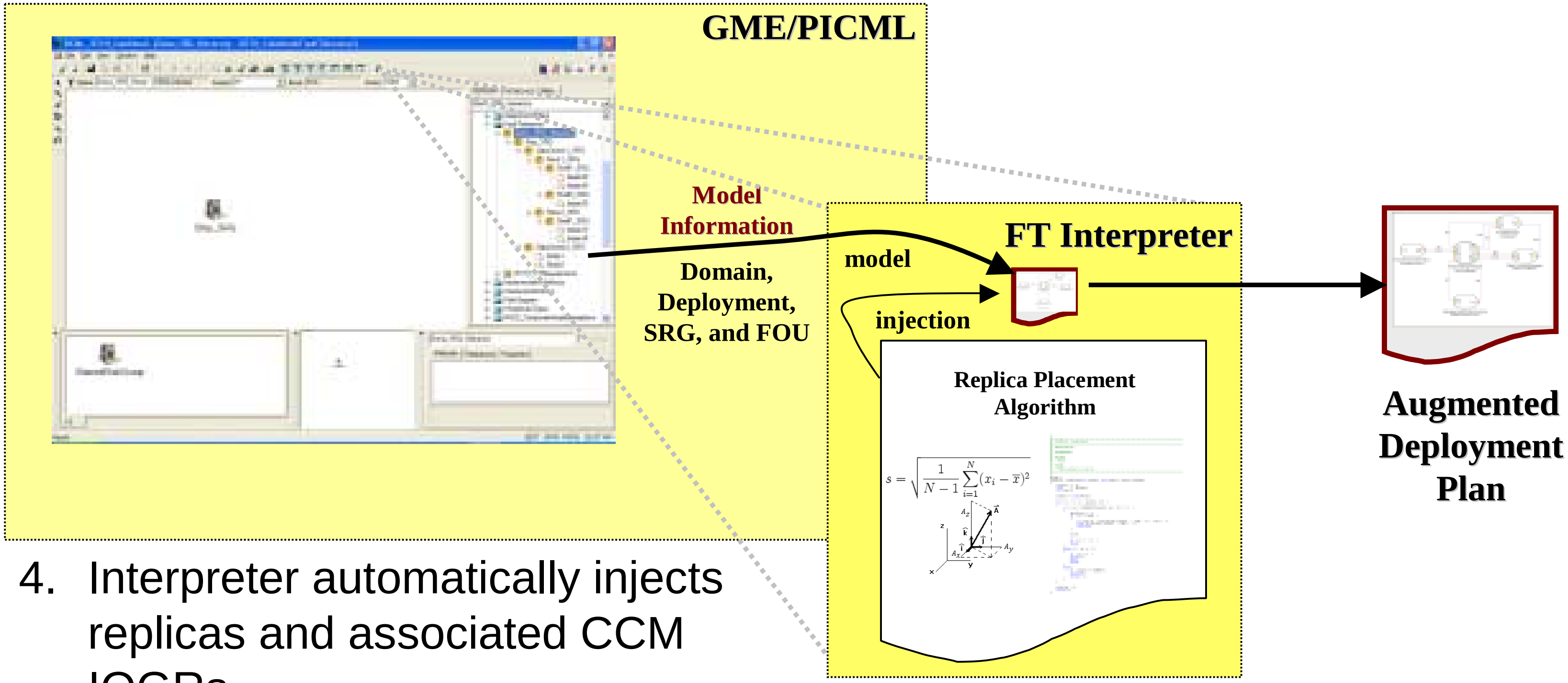
$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Component Placement Example using SRGs



FT Modeling & Generative Steps

1. Model components and application strings in PICML
2. Model Fail Over Units (FOUs) and Shared Risk Groups (SRGs)
3. Generate deployment plan



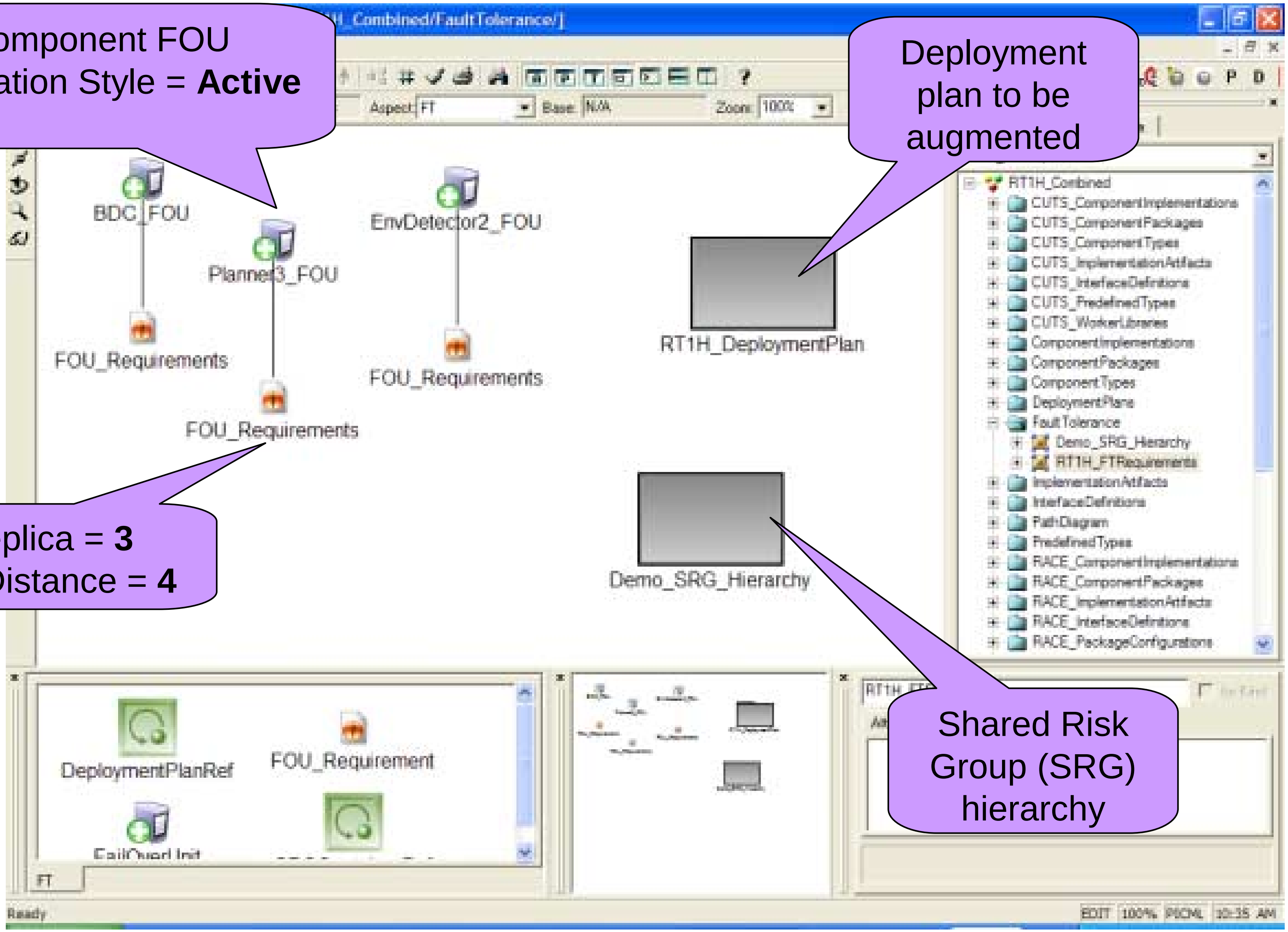
4. Interpreter automatically injects replicas and associated CCM IOGRs
5. Distance-based constraint algorithm determines replica placement in deployment descriptors.

FT Requirements Screenshot in CoSMIC (PICML)

LEGEND

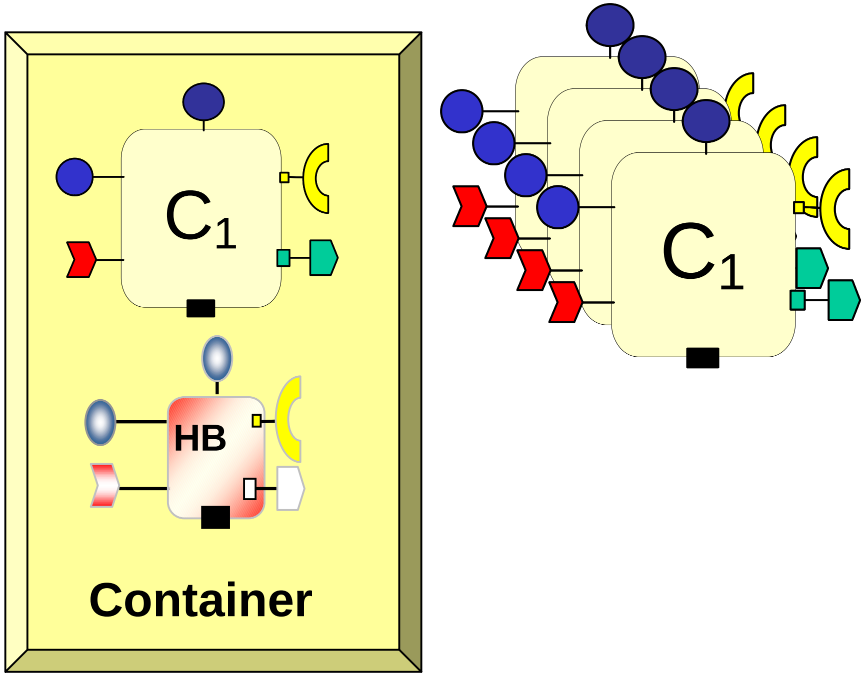
FOU: Fail Over Unit

SRG: Shared Risk Group

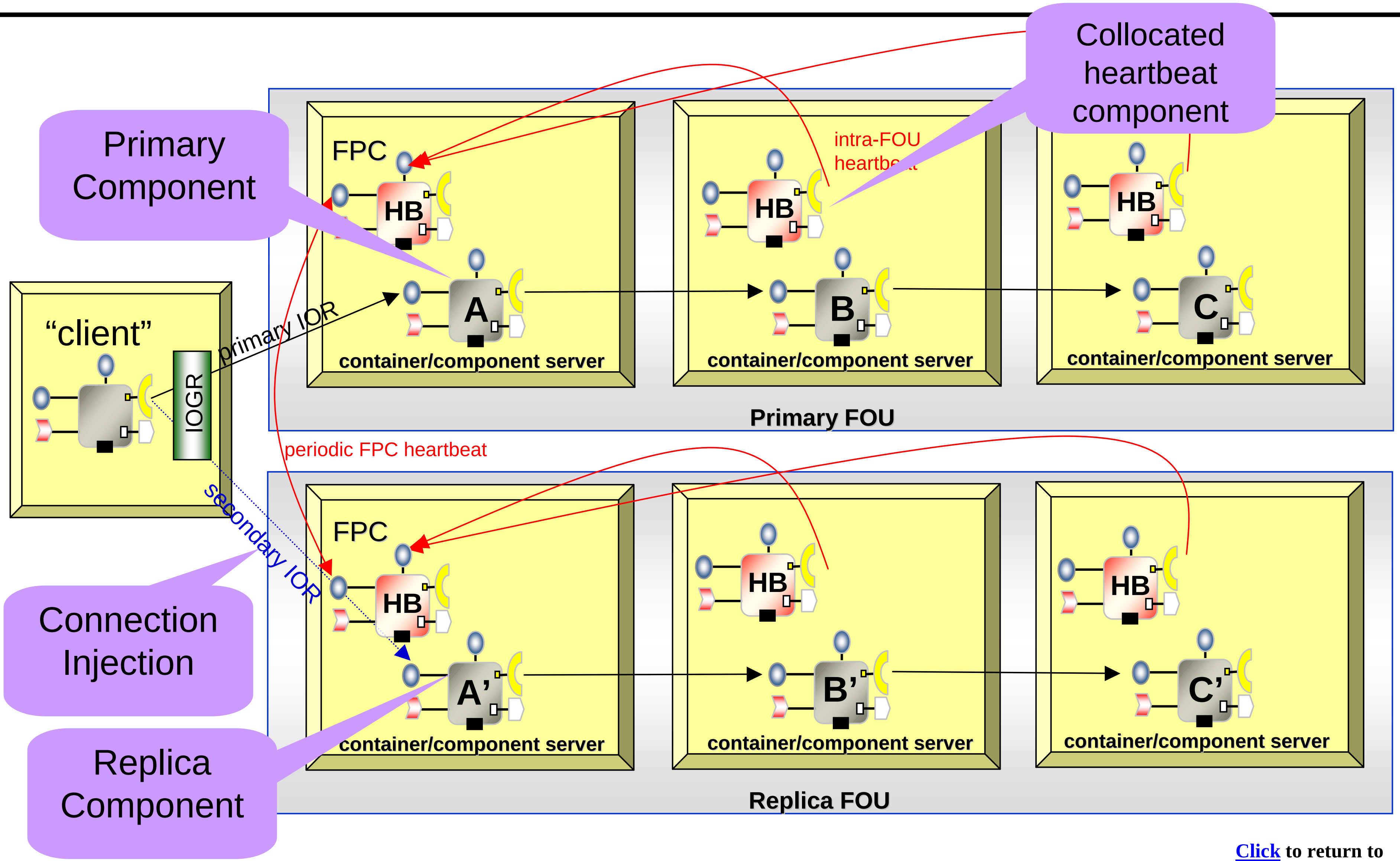


Generative Capabilities for Provisioning FT

- Automatic Injection of replicas
  - Augmentation of deployment plan based on number of replicas
- Automatic Injection of FT infrastructure components
  - E.g. Collocated "heartbeat" (HB) component with every protected component.
- Automatic Injection of connection meta-data
  - Specialized connection setup for protected components (e.g. Interoperable Group References IOGR)



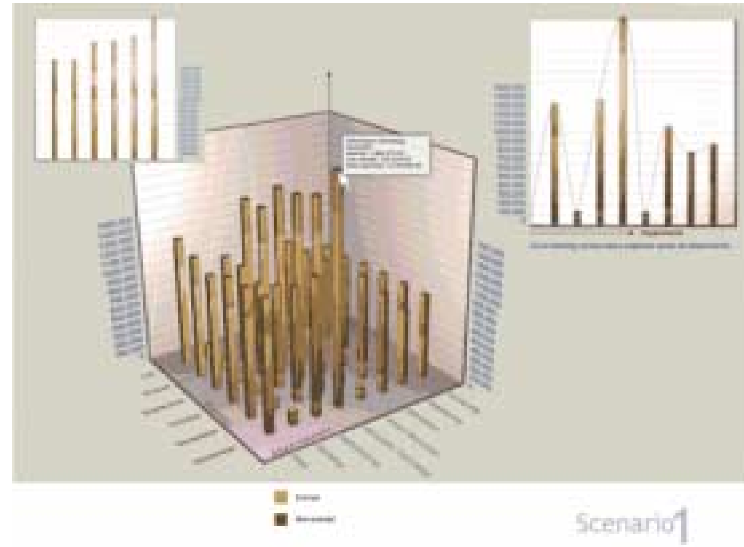
Example of Automated Heartbeat Component Injection

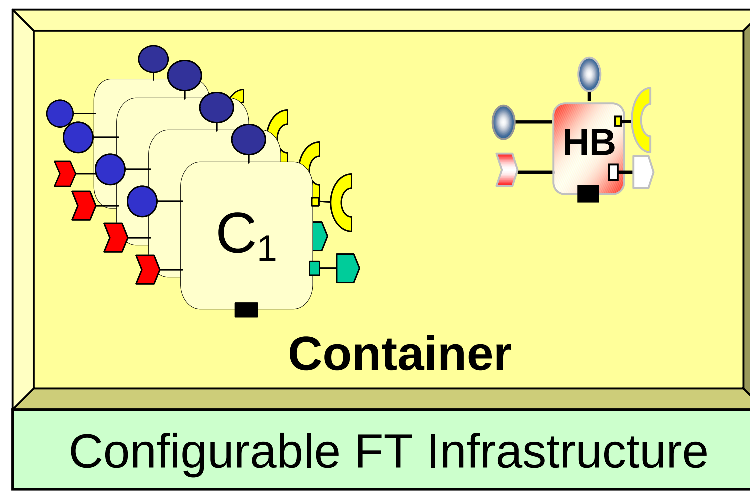


Click to return to concepts slide.

Future Work

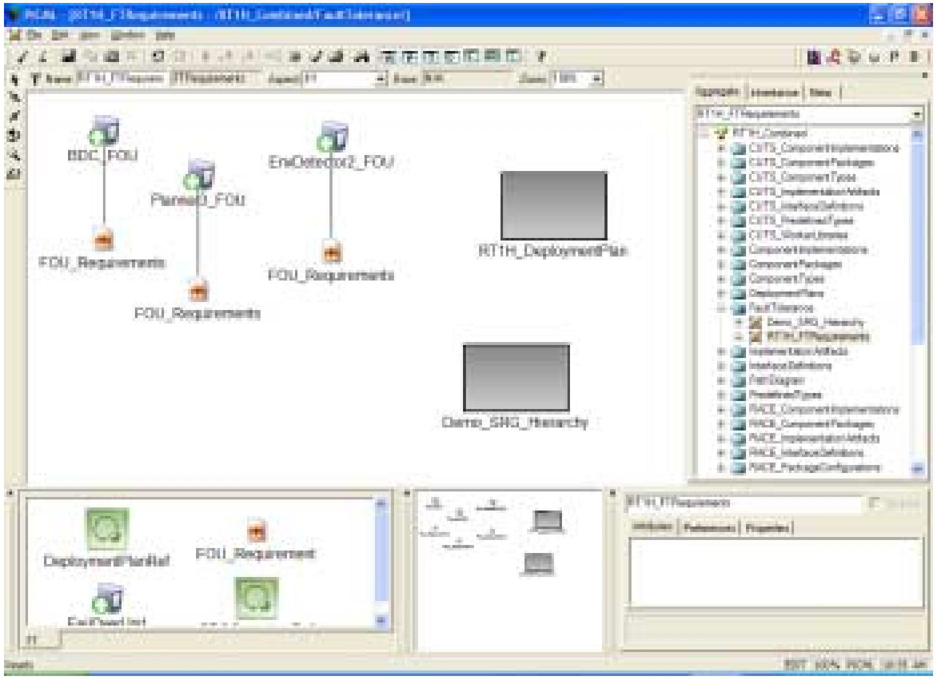
- Developing advanced constraint solver algorithms to incorporate multiple dimensions of constraints in component placement decision (e.g. resources, communication latency)
- Optimizing the number of generated heartbeat components for collocated, protected application components.
- Enhancing the DSL and the tools to capture the configurability required by the new Lightweight RT/FT CORBA specification.
  - E.g. Enhancing the model interpreter to support a wide spectrum of established fault-tolerance mechanisms
- Enhancing working prototypes and evaluating them in representative DRE systems





Concluding Remarks

- Model-Driven Engineering separates dependability concerns from other system development concerns
- Separation of concerns helps alleviate system complexity
- Model-based generative capabilities “compile” FT infrastructure (e.g. heartbeat components and connections) during model interpretation time and synthesize meta-data



Tools available for download from
www.dre.vanderbilt.edu/cosmic
www.dre.vanderbilt.edu/CIAO