

Applying OMG Model Driven Architecture to Distributed Real-time and Embedded Systems

Aniruddha Gokhale (a.gokhale@vanderbilt.edu)
Douglas C. Schmidt (schmidt@uci.edu)
Balachandran Natarajan (bala@cs.wustl.edu)
Nanbor Wang (nanbor@cs.wustl.edu)

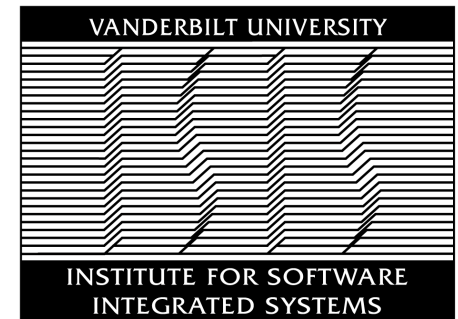
deuce.doc.wustl.edu/CoSMIC



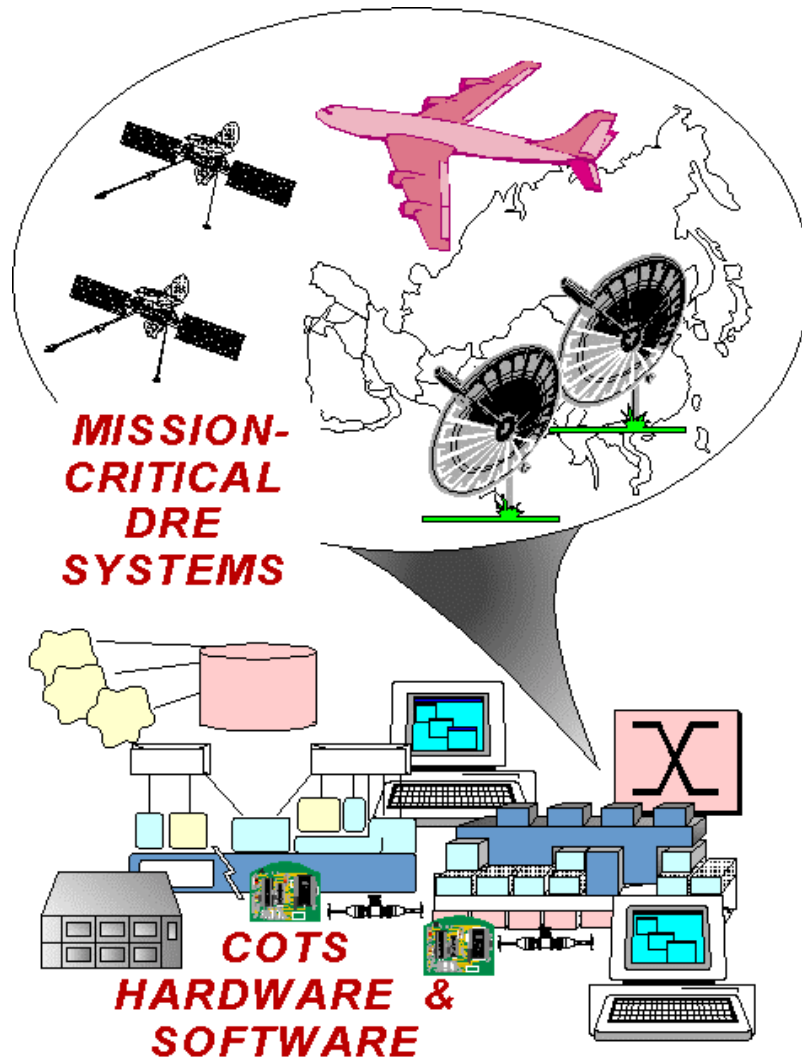
Presented at

OMG Distributed Real-time Systems Workshop

July 15-18, 2002, Arlington, VA



Motivating Technology Forces



Context

- DRE systems increasingly built from COTS h/w and s/w

Problem

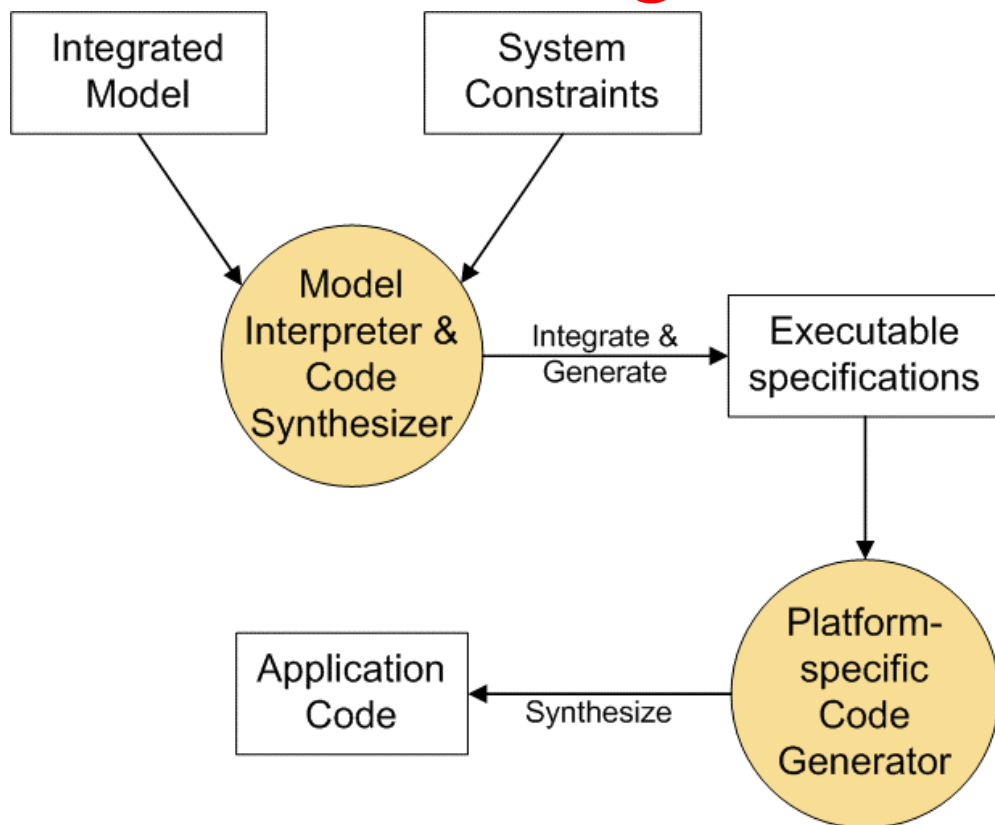
- Proliferation of middleware technologies – *CORBA, Java RMI, COM+*
- Satisfying multiple QoS service requirements – no one-size-fits all
- Accidental complexities assembling, integrating and deploying software systems

Solution

- Integrate model integrated computing (MIC) with QoS-enabled component middleware
- Integrate MIC with Web services



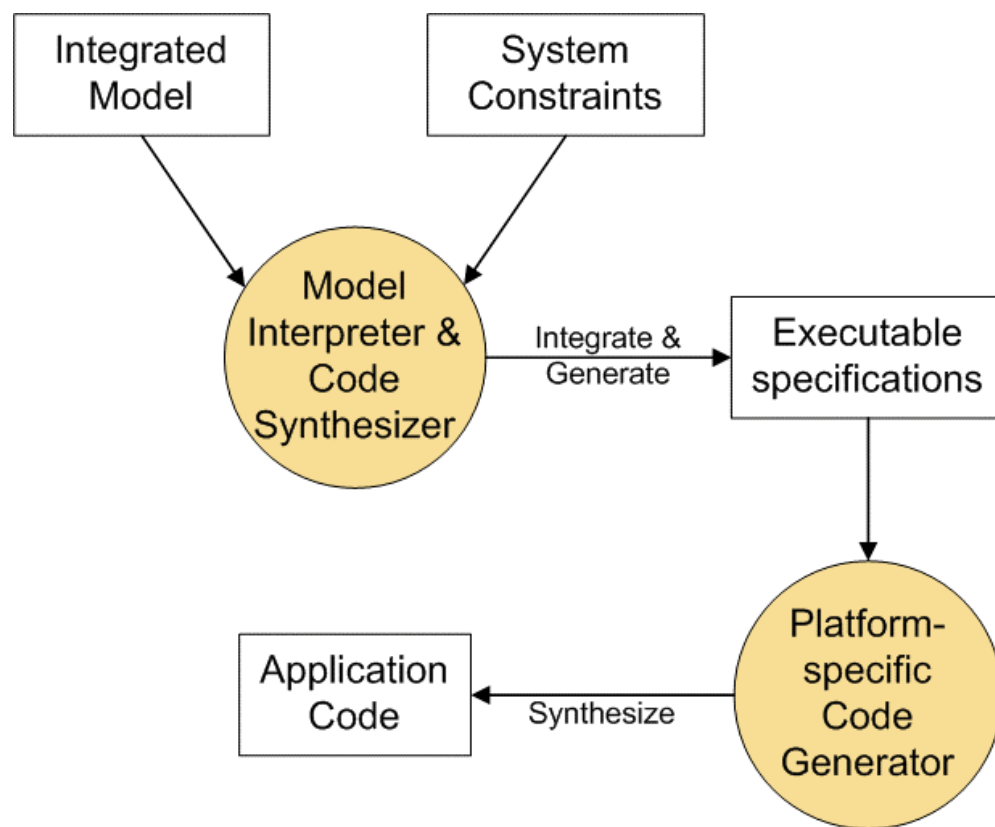
Model Integrated Computing (MIC)



- **Analyze** – different but interdependent characteristics of DRE system behavior
- **Synthesize** – platform-specific code customized for DRE application

- Applies domain-specific modeling languages to engineer computing systems
- Provides rich modeling environment including model analysis and model-based program synthesis
- Modeling of integrated end-to-end view of applications with interdependencies
- Captures the essence of a class of applications
- Modeling languages and environments themselves can be modeled as meta-models
- *e.g.*, Generic Modeling Environment (GME) (www.isis.vanderbilt.edu)

Model Integrated Computing (MIC)



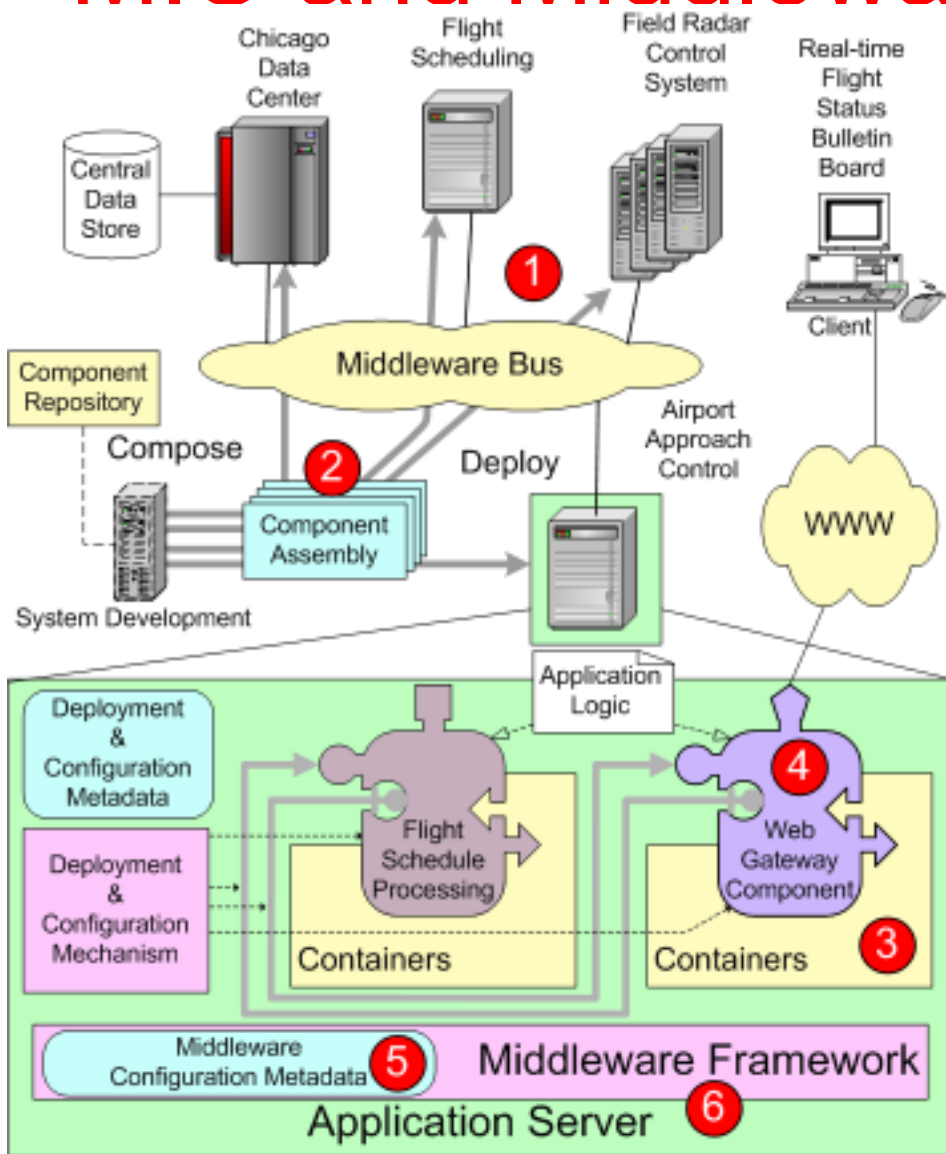
Advantages

- Free application developers from dependencies on any particular API
- Analyze models and provide correctness proofs
- Highly dependable and robust synthesized code
- Rapid prototyping of new concepts via modeling and interpreting
- Reducing time-to-market and saving costs, preserving investments
- Resolve interoperability issues by synthesizing standard or custom code

www.isis.vanderbilt.edu



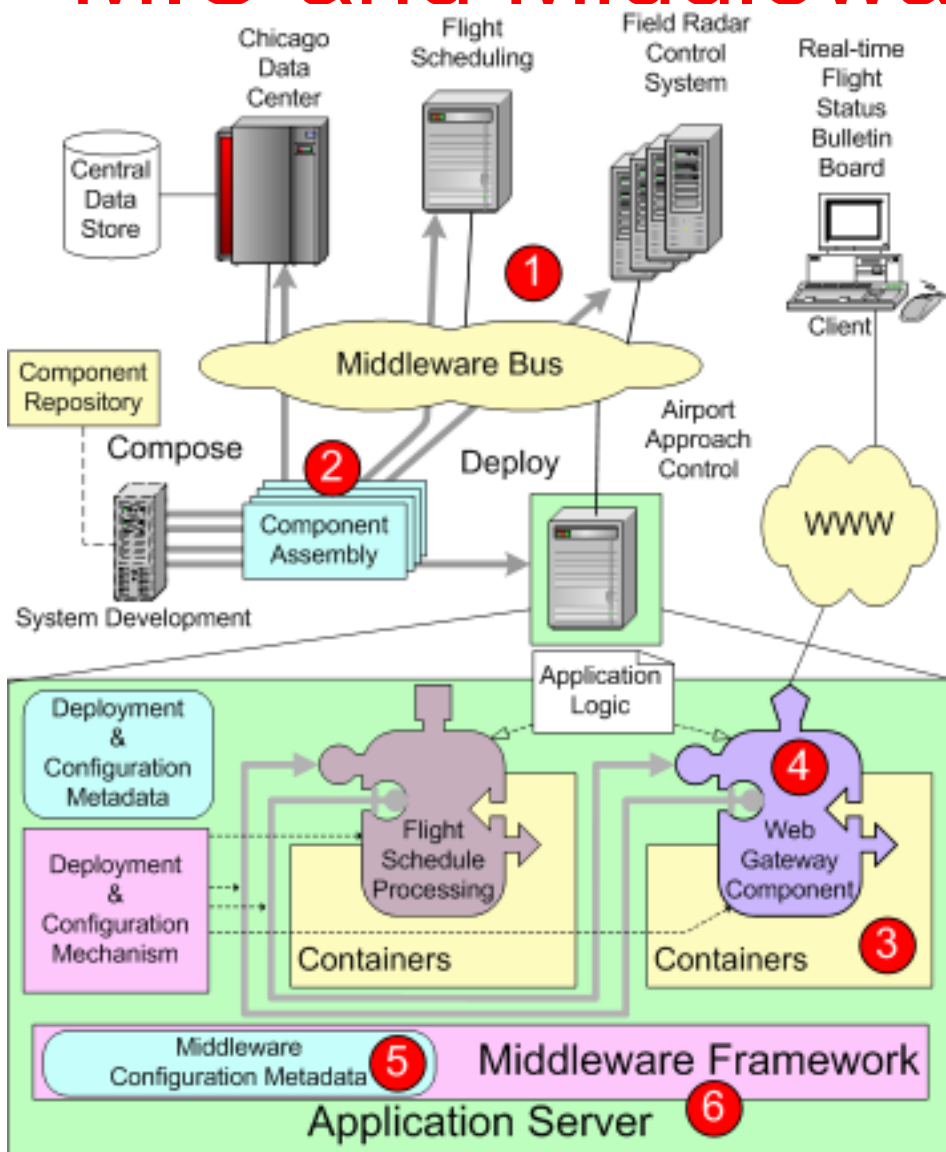
MIC and Middleware/Web Integration



1. Configuring and deploying application services end-to-end
 - partitioning and distributing
 - provisioning resources for QoS
2. Composing components into application servers
 - assemble semantically compatible QoS-enabled components from reuse repositories
 - determining interconnections between components in metadata
 - packaging components and metadata
3. Configuring application component containers
 - configuring right QoS policies for component containers
 - maintain inter-component semantic compatibility w.r.t container policies



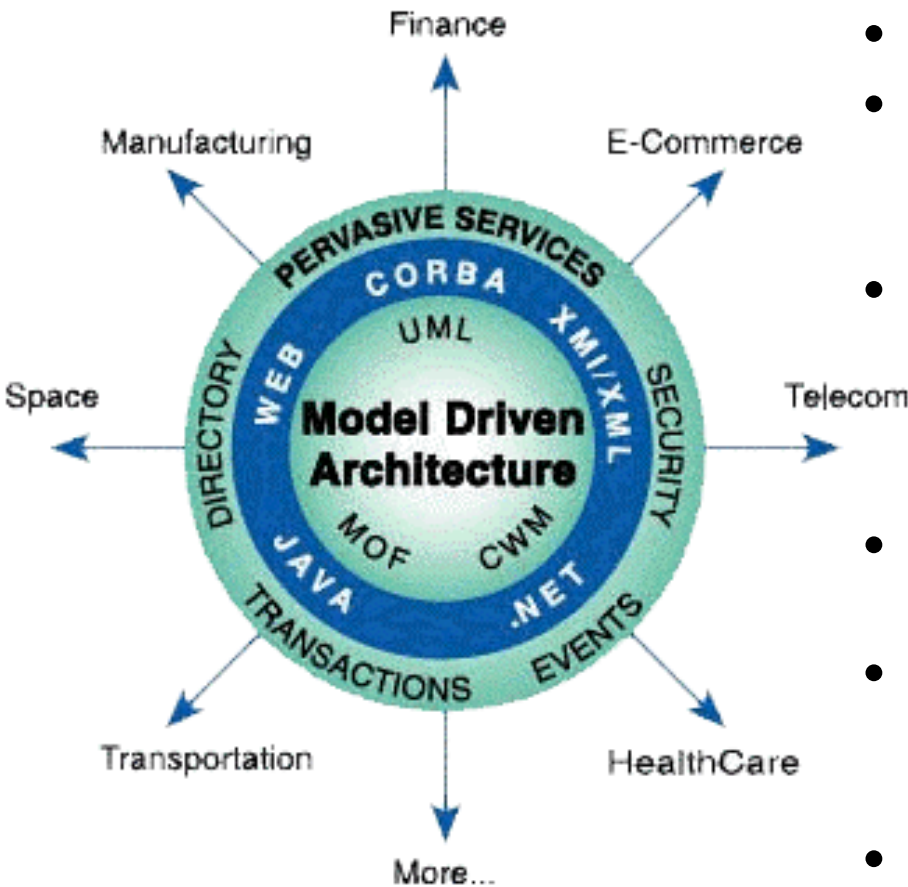
MIC and Middleware/Web Integration



4. Synthesizing application component implementations
 - synthesis of DRE components tailored to application e.g., for bounded worst case execution time under overload conditions
 - bridge the gap between specifications and implementation via aspect weavers and generators
5. Synthesizing middleware-specific configurations
 - configuring threading models, buffering and flow control, levels of fault tolerance, transport protocols, demultiplexing strategies, security
6. Synthesizing middleware implementations
 - a more aggressive approach to synthesizing custom middleware



OMG Model Driven Architecture

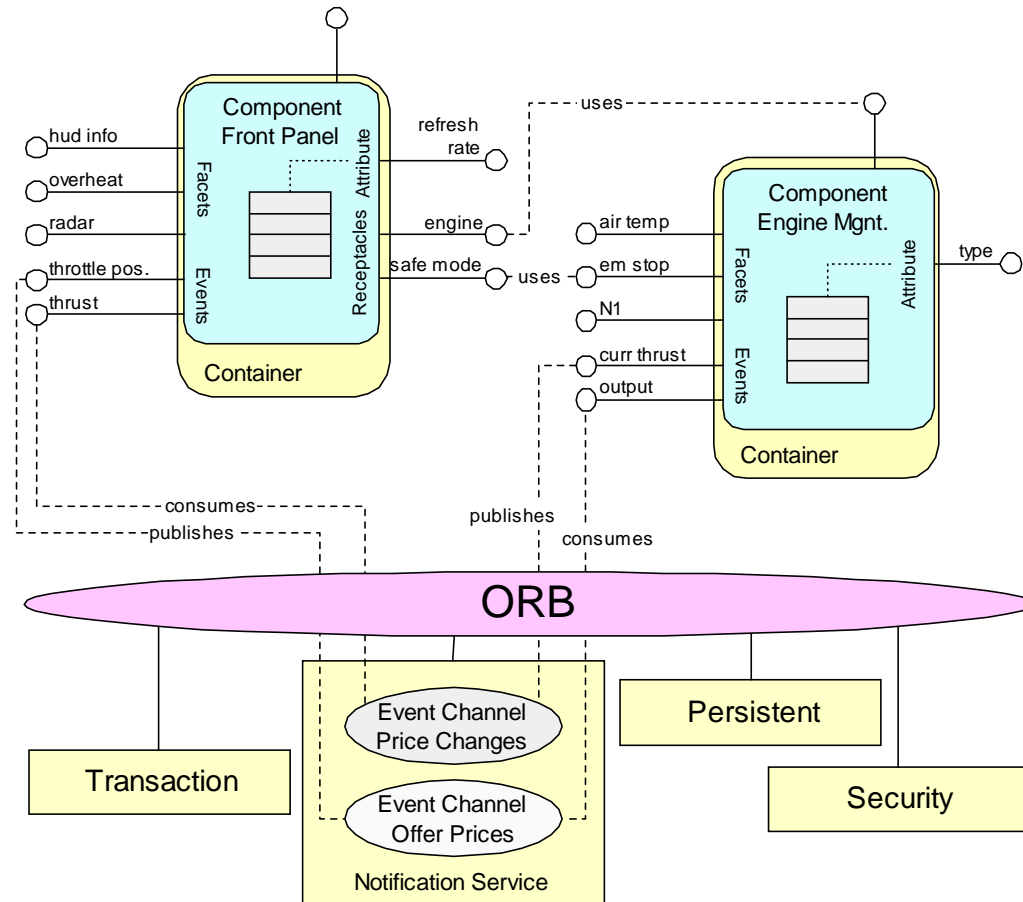


- Standardization of the MIC technology
- Defines platform- independent models (PIMS) and platform-specific Models (PSMs)
- Uses Unified Modeling Language (UML) for modeling
 - real-time profile
 - dynamic scheduling profile
- Meta Object Facility (MoF) serves as meta-model repository
- Common Warehouse Model (CWM) provides standard interfaces to manage different databases
- XML Metadata Interchange (XMI) for meta-model exchange
- 3 levels of specifications – pervasive services, domain facilities, applications

www.omg.org/mda

Component-Integrated ACE ORB (CIAO)

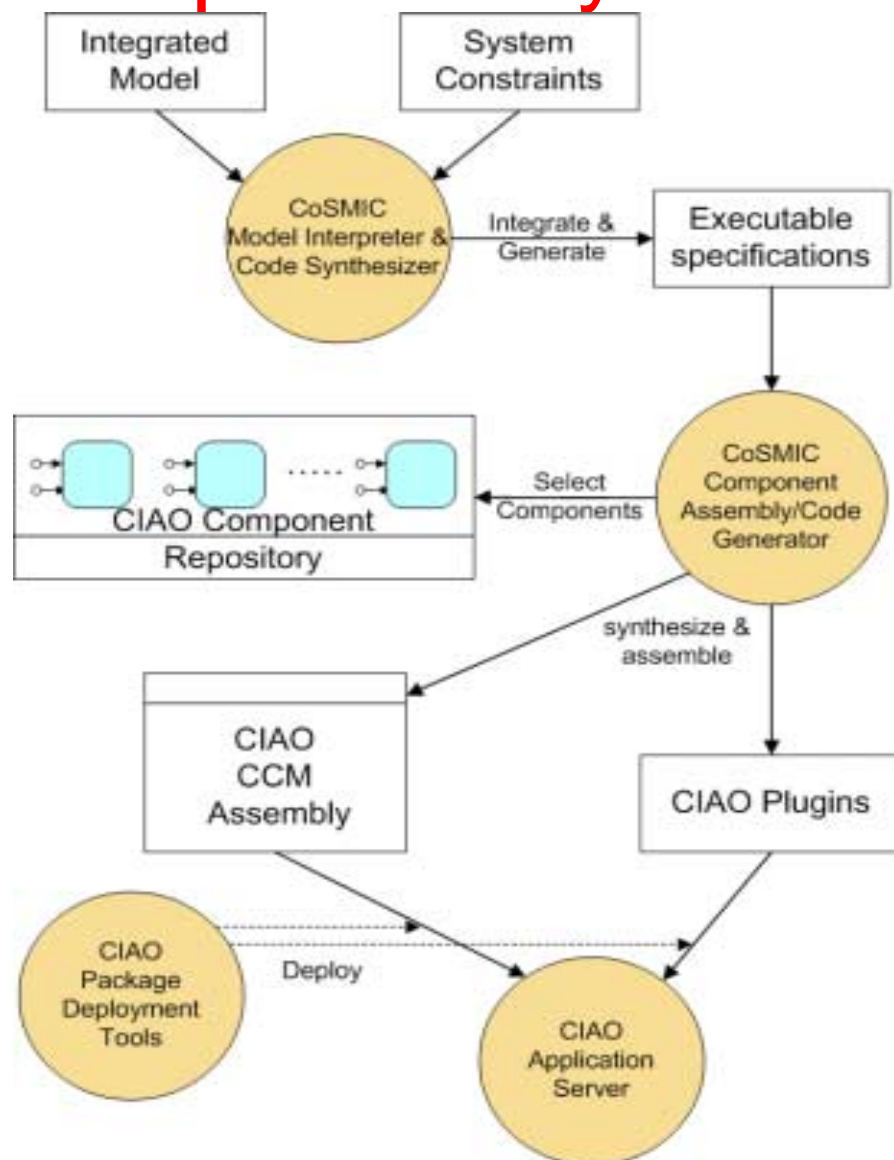
CCM incarnation of TAO ORB



- Support development via composition
 - providing CCM framework
- QoS-aware
 - decouple QoS policies specification from component implementations
 - specify QoS policies in component assembly descriptors
- Configurable
 - leveraging hardware capabilities
 - composing QoS supporting mechanisms for CCM application servers

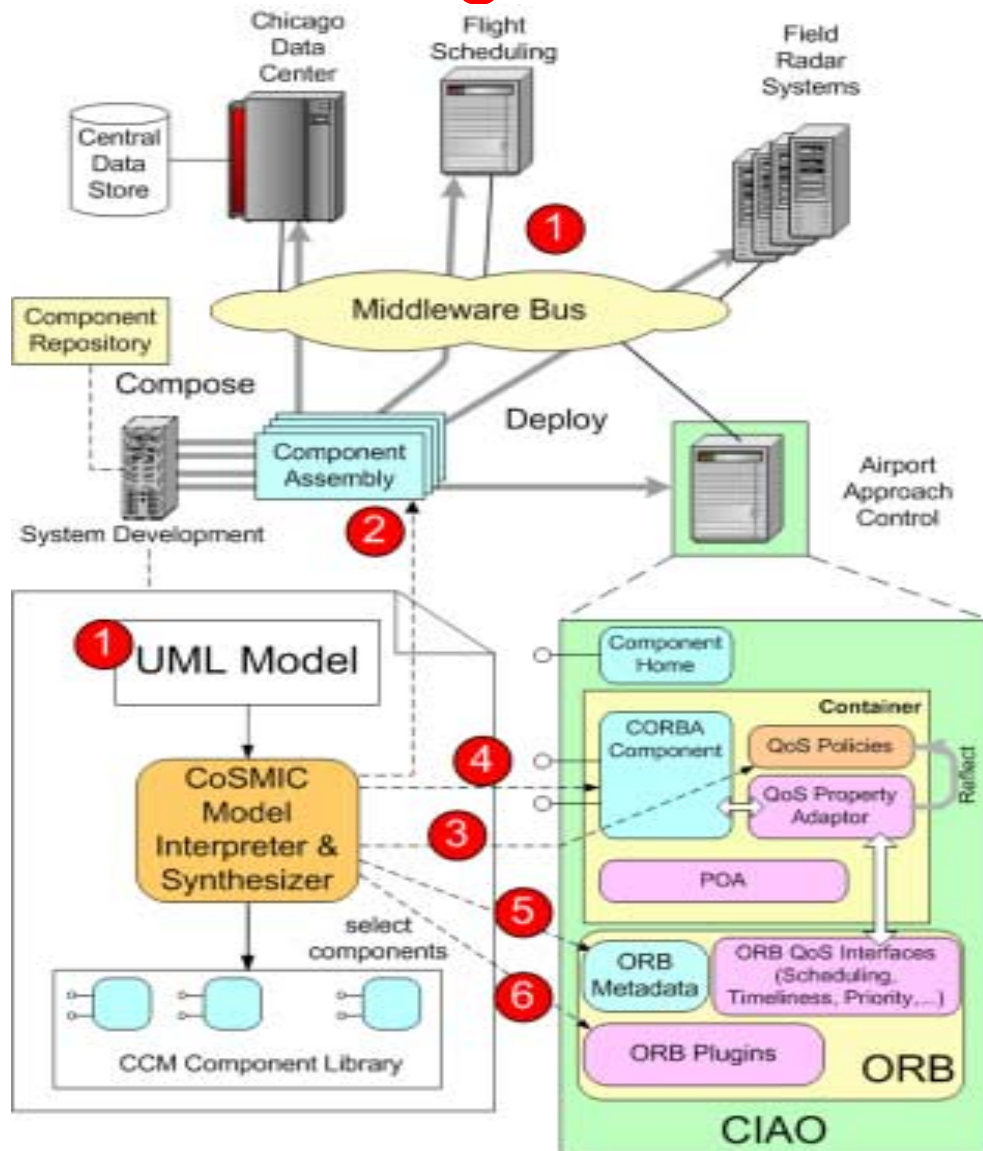
deuce.doc.wustl.edu/CIAO

Component Synthesis with MIC (CoSMIC)



- Synthesizes code and configuration metadata for the CIAO CORBA Component middleware
 - reusing components via compositions over generating new component implementations
 - composition of applications components & CIAO plug-ins
 - CIAO helps instantiating application processes
- MDA tool suite
 - UML modeling using GME
 - analysis and synthesis tools
- Enhancement to GME tool
 - uses MDA standards-based approach

Resolving DRE Challenges with CoSMIC



Proliferation of middleware

- UML modeling tools used to model DRE application behavior – points (1) and (2)
- model-first/generate-next strategy for finer grained control in components shown in point (4)

Simultaneous support for multiple QoS

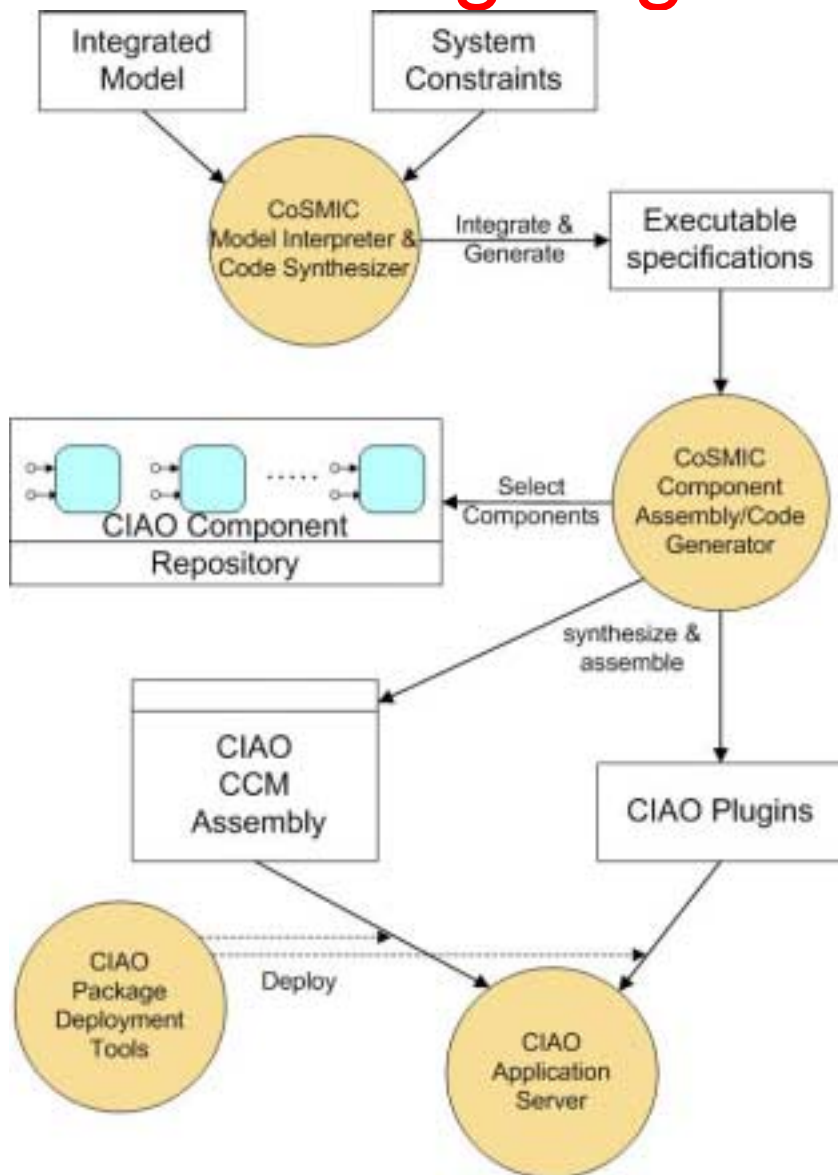
- model overall application QoS and partitioning shown in point (1)
- compose application servers shown in point (2)
- model and synthesize components shown in point (4)
- validate and deploy – points (3, 5)

Accidental Complexities

- container QoS configuration policies, metadata – point (3)



Ongoing Work on CoSMIC



GME Enhancements

- porting to other platforms using Java
- addition of MoF capabilities

Design Space Exploration Tools

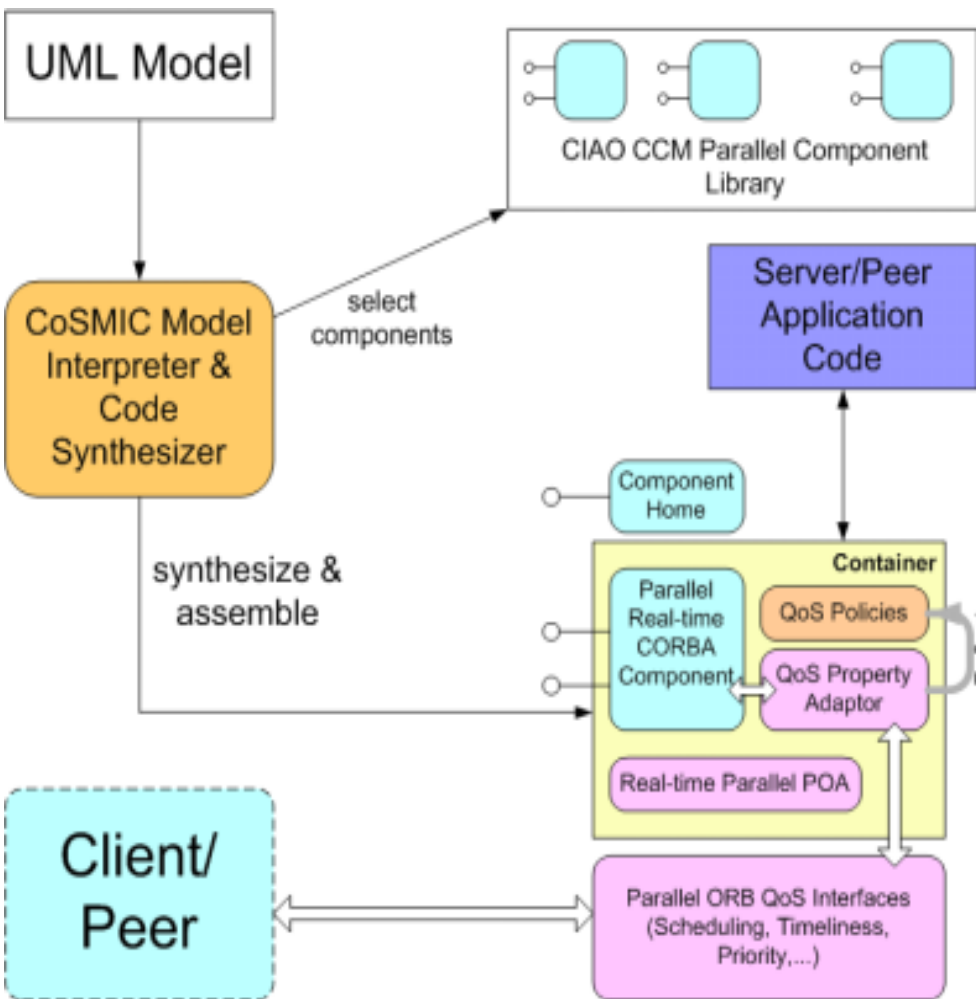
- navigating runtime adaptation space
- integrating with BBN QuO

Analysis and Synthesis Tools

- component composition
- XML configuration metadata generation



Future Work



Integration with Web services

- exposing synthesized component middleware as web service
- synthesis of code to register with UDDI

Modeling and Synthesis for GRID Applications

- composition of Data Parallel CORBA components and integration with CIAO RTCCM
- model driven service and resource provisioning

