Model Driven Middleware

Presented by:

Thomas Repantis

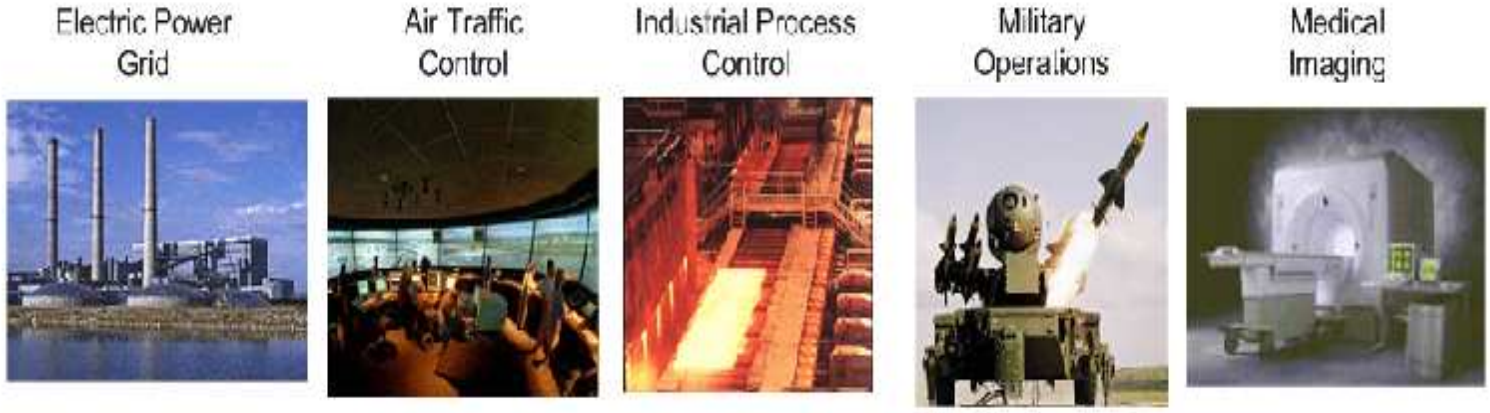
trep@cs.ucr.edu

Overview

Applying Model Driven Architectures to Distributed Systems.

1. Distributed Real-Time Embedded Systems
2. Component Middleware
3. Model Driven Middleware
4. Existing Work on Real-Time MDM
5. Our Vision for Fault-Tolerant, Secure MDM

Distributed Real-Time Embedded (DRE) Systems



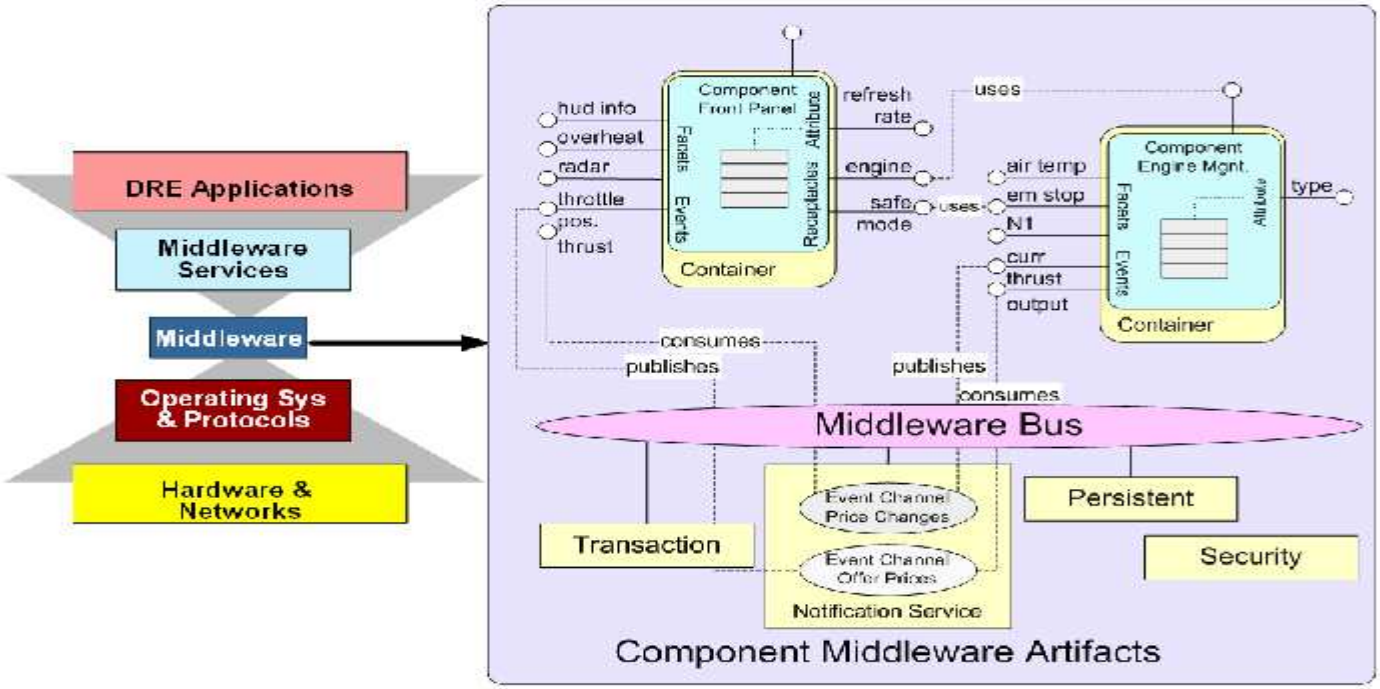
Networks:

- Large-scale
- Heterogeneous
- Dynamic

DRE Requirements

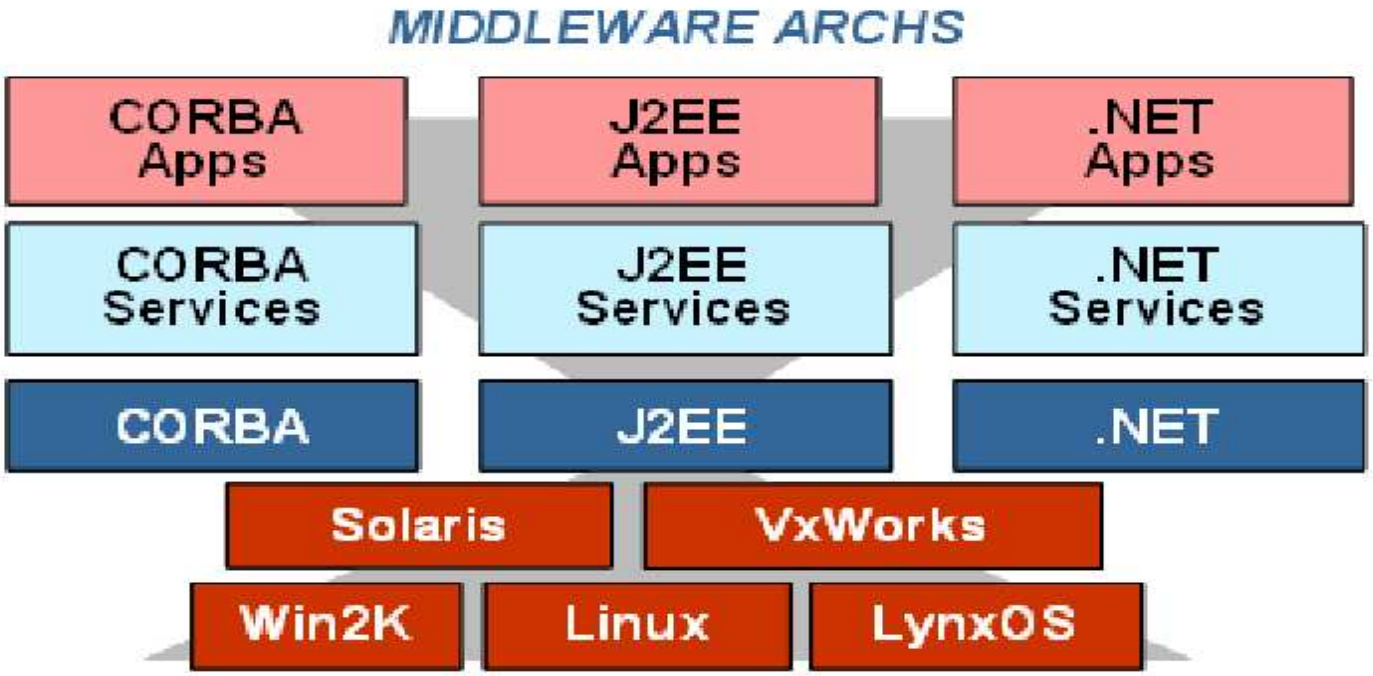
- **Real-time**:
  - low latency, bounded jitter
- **Availability**:
  - bounded fault propagation/recovery
- **Security**:
  - authentication, authorization
- **Physical Requirements**:
  - weight, power consumption, memory footprint

Component Middleware



- Control of QoS properties
- Platform independence
- Cost reduction

Middleware Architectures



- Real-time CORBA
- Fault-tolerant CORBA

Unresolved Challenges

- Isolation of applications from middleware platforms
- Composing applications from components
- Configuring component middleware
- Automated deployment
- Satisfying multiple QoS properties simultaneously

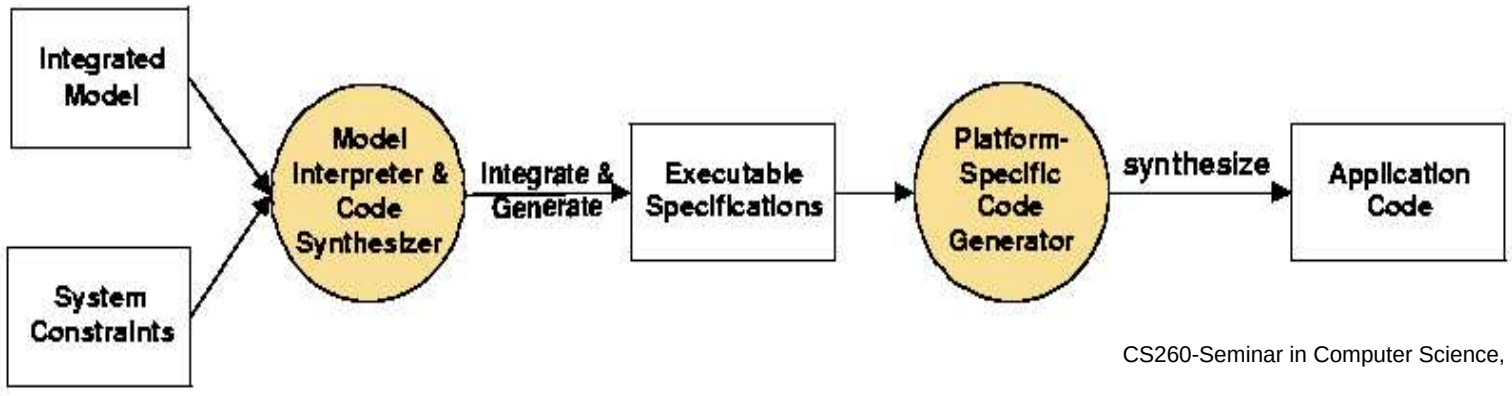
Ad hoc (manual) techniques:

- Do not scale well
- Are tedious
- Are error-prone
- Lack verification and validation mechanisms

MDA to the rescue...

MDA can express application functionality and QoS requirements at higher levels of abstraction than by using 3GLs:

- Model properties
- Analyze requirements
- Synthesize code
- Provision deployment

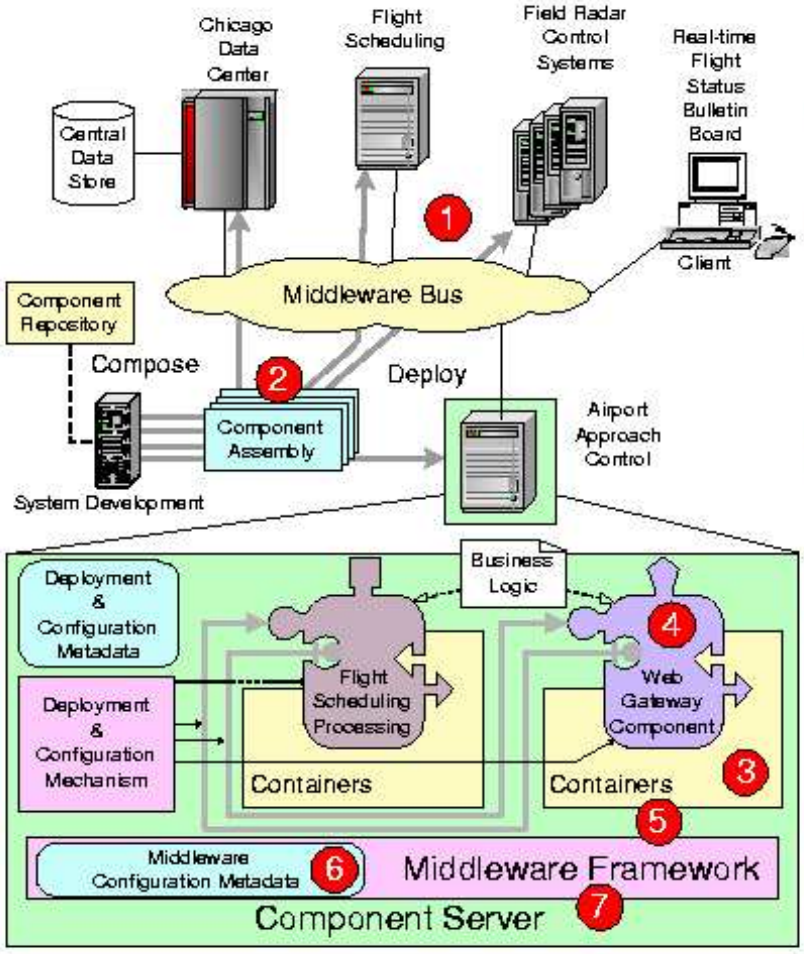


Model Driven Middleware

Bridge the gap between specification and implementation:

- Compose applications from reusable components.
- Synthesize new extended components.
- Automate the configuration of QoS aspects.
- Model the interfaces of components in a standard way.
- Easily handle changes in components.

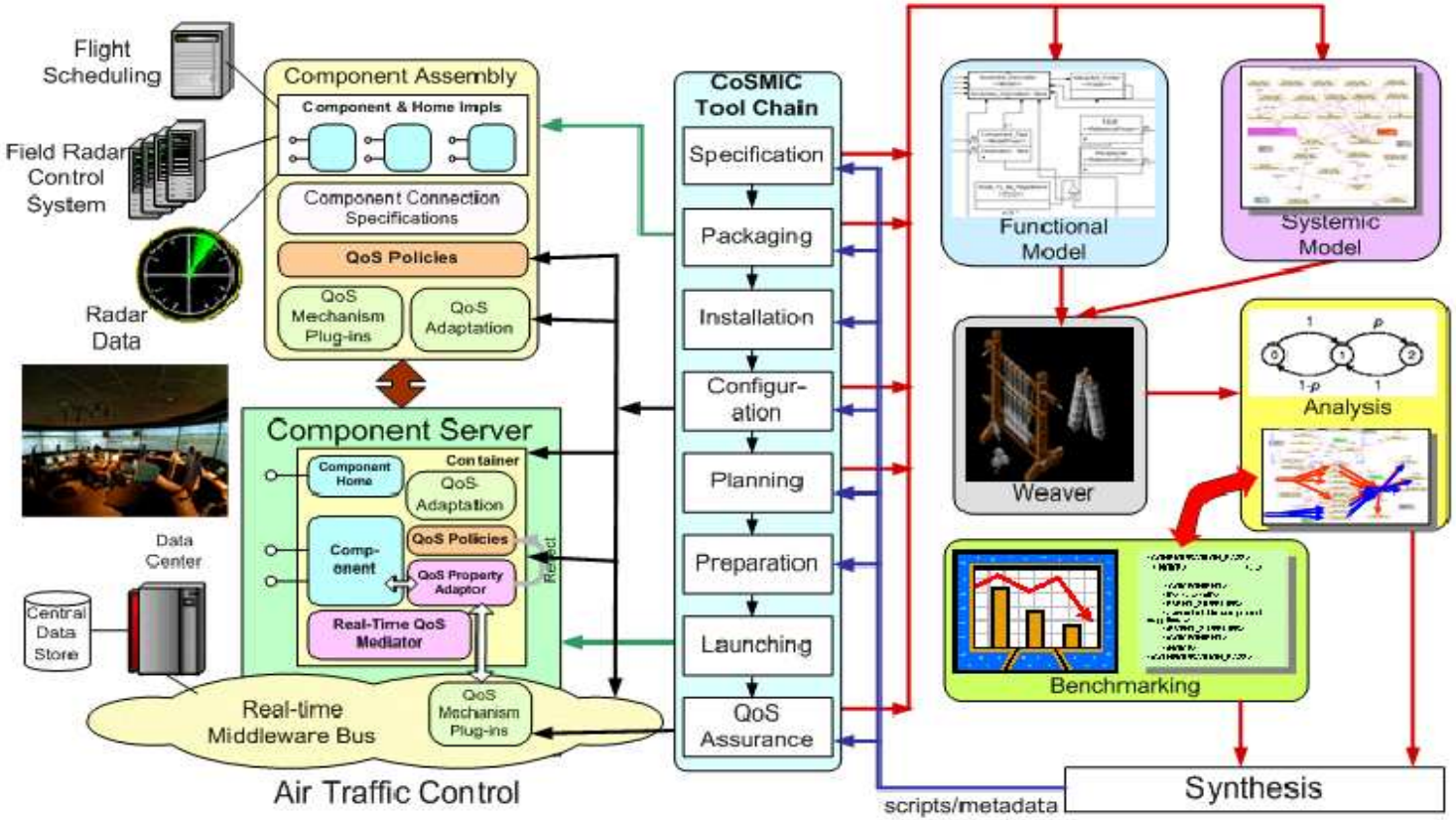
MDM Example



1. Configuring and deploying an application services end-to-end
2. Composing components into application server components
3. Configuring application component containers
4. Synthesizing application component implementations
5. Synthesizing dynamic QoS provisioning and adaptation logic
6. Synthesizing middleware-specific configurations
7. Synthesizing middleware implementations

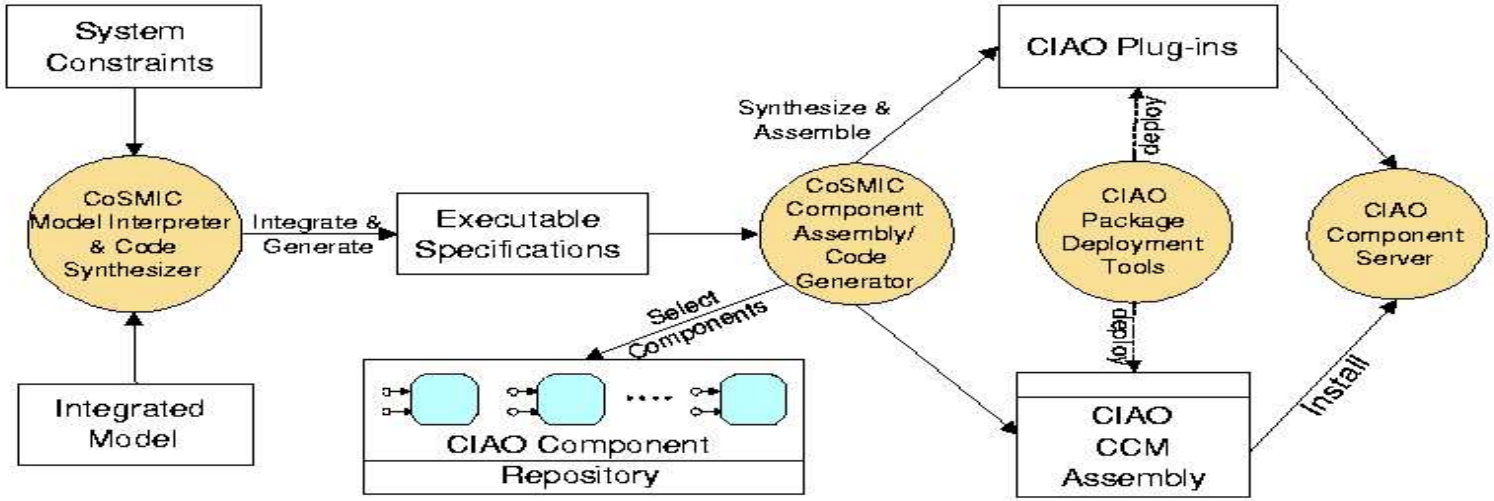
Existing Work on Real-Time MDM

Component Synthesis using Model Integrated Computing (CoSMIC - Douglas Schmidt - Vanderbilt University)



CoSMIC

- Each CoSMIC tool synthesizes metadata in XML for use in the underlying middleware.
- CoSMIC uses a Platform Specific Model to integrate the modeling technology with the CIAO QoS-enabled component middleware.



CoSMIC Details

CoSMIC Modeling Paradigms:

- OCML (Options Configuration Modeling Language) to model configuration parameters and constraints and synthesize the middleware configuration metadata.
- CADML (Component Assembly and Deployment Modeling Language) to model component assembly and deployment.

Our Vision for Fault-Tolerant, Secure MDM

Systems able to continue normal operation despite the presence of hardware or software faults:

- Communication network failures
- Node failures
- Object failures

Different security levels and domains:

- Authentication
- Authorization

Goals

Fault-tolerance:

- Automatic creation and allocation of replicas
- Automatic maintenance of replica consistency
- Automatic fault detection and recovery

Security:

- Automatic admission control
- Automatic conformance to specific security levels

System Architecture

- Replication Manager
- Fault Detector
- Admission Manager

System Parameters:

- Probability of failure for each component
- Replication degree of each component
- Security level satisfied by each component
- Access privileges of each component

A Model-Driven Approach

- Modeling of components
- Configuration of parameters
- Deployment
- Fault-tolerance and security assurance

Conclusions

- Distributed Real-Time Embedded Systems are increasingly being developed using component middleware.
- Unresolved challenges include isolation of applications from the middleware platform, automatic application composition, and automatic middleware configuration.
- Model Driven Architectures can provide a scalable and verifiable solution to the above.
- Model Driven Middleware can automate the creation, configuration, and deployment of Real-Time, Fault-Tolerant, Secure distributed applications.

References

1. Aniruddha Gokhale, Douglas C. Schmidt, Balachandran Natarajan, Jeff Gray, and Nanbor Wang, “Model Driven Middleware”, Middleware for Communications, Wiley and Sons, 2003.
2. Aniruddha Gokhale et al., “ Model Driven Middleware: A New Paradigm for Deploying and Provisioning Distributed Real-time and Embedded Applications”, Elsevier Journal of Science of Computer Programming: Special Issue on Model Driven Architecture, 2004.
3. Aniruddha Gokhale et al., “CoSMIC: An MDA Generative Tool for Distributed Real-time and Embedded Applications”, Workshop on Model-driven Approaches to Middleware Applications Development at 4th IFIP/ACM/USENIX International Conference on Middleware for Distributed Systems Platforms, 2003.
4. The OMG Real-Time CORBA Specifi cation v1.1, 2002.
5. The OMG Fault Tolerant CORBA Specifi cation v1.0, 2000.
6. The OMG MDA Guide v1.0.1, 2003.

Thank you!

Questions/Comments?