Model-based Design and Network Centric Systems

Janos Sztipanovits
ISIS, Vanderbilt University

DATE 2006
Munich, Germany
March 6, 2006

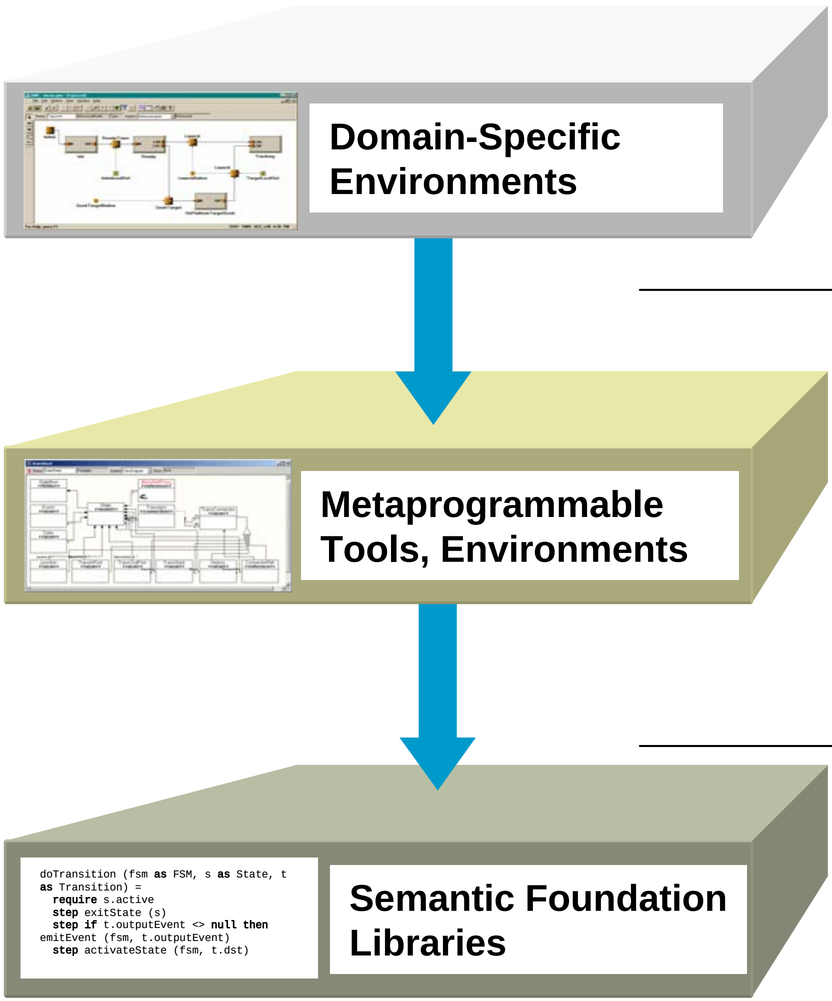
Content

- Model Based Design and MIC
  - Modeling
  - Model Data Management
  - Model Transformation
  - Tool Integration
- Modeling in dynamic architectures
- Modeling in sensor network applications

Goal and Approaches

- Building increasingly complex networked embedded systems from components
  - Naïve "plug-and-play" approach does not work in embedded systems (neither in larger non-embedded systems)
  - Model-based software design focuses on the *formal representation, composition, analysis and manipulation of models* during the design process.
- Approaches with differences in focus and details
  - MDA: Model Driven Architecture
  - MDD: Model-Driven Design
  - MDE: Model-Driven Engineering
  - **MIC: Model-Integrated Computing**

Model-Integrated Computing Approach (ISIS-VU)



Modeling Domain Specific Design Flows: Examples in MIC:

- ECSL - Automotive
- ESML - Avionics
- SPML - Signal Processing
- CAPE/eLMS – Learning Technology
- AADL….

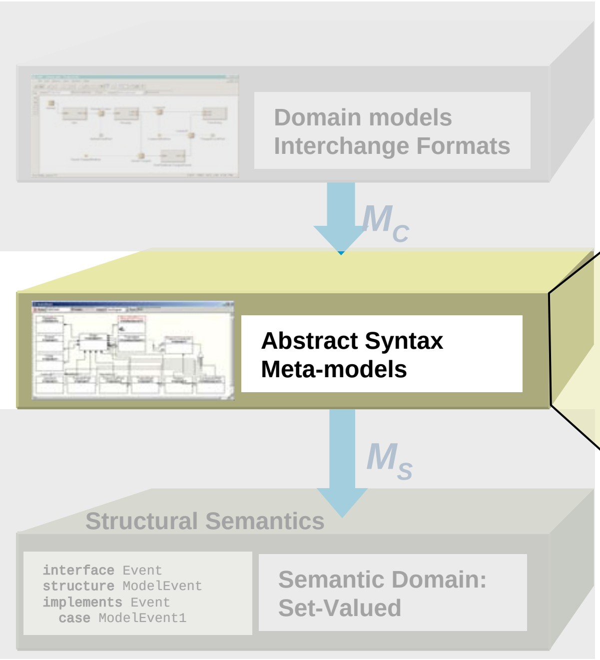
Metamodeling and Metaprogrammable Tools: (mature or in maturation program)

- GME (Generic Model Editor)
- GReAT (Model Transformation)
- OTIF (Tool Integration Framework)
- UDM (Universal Data Model)
- DESERT (Design Space Exploration)

Modeling Semantics (work in progress):

- Semantic "Units"
- Semantic Anchoring

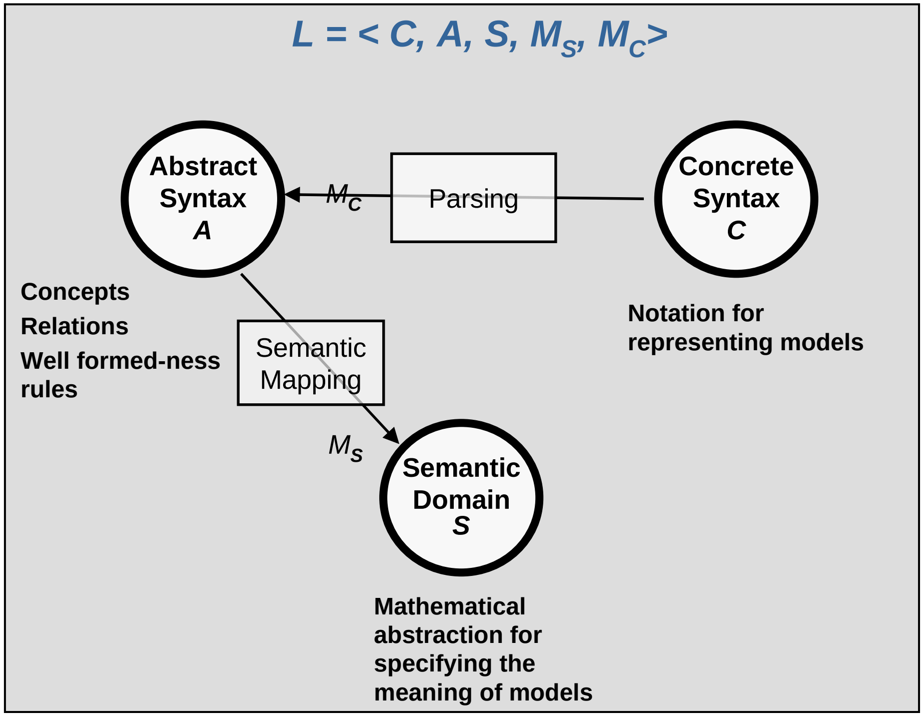
Metamodeling Layer Objectives



- Metamodeling
- Model Data Management
- Model Transformation
- Tool Integration

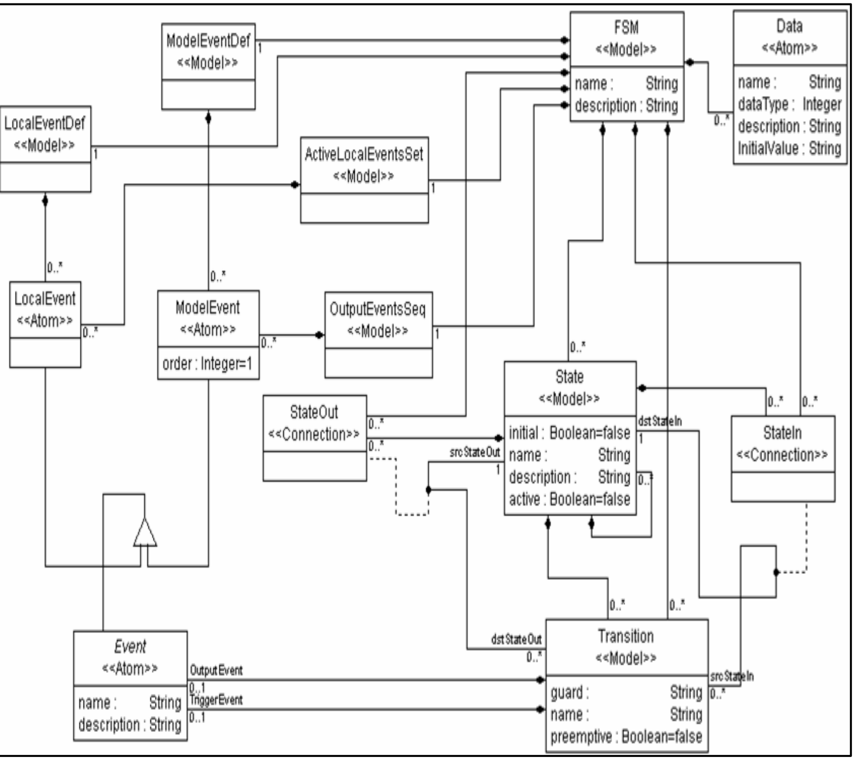
Metamodeling and Domain Specific Modeling Languages

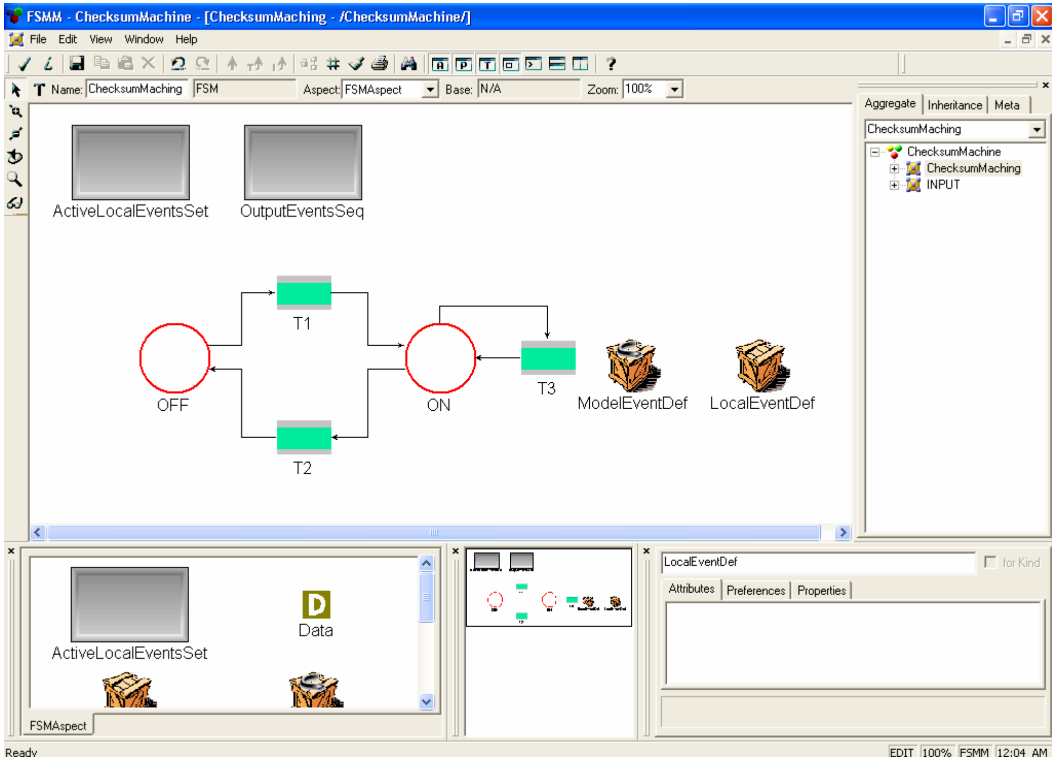
Domain Specific Modeling Language (DSML)



- **Model**: precise representation of artifacts in a modeling language L
- **Modeling language**: defined by the notation (C), concepts/relations and integrity constraints (A), the semantic domain (S) and mapping among these.
- **Metamodel:** formal (i.e. precise) representation of the modeling language L using a metamodeling language L_M.

Modeling Example: Metamodel and Models





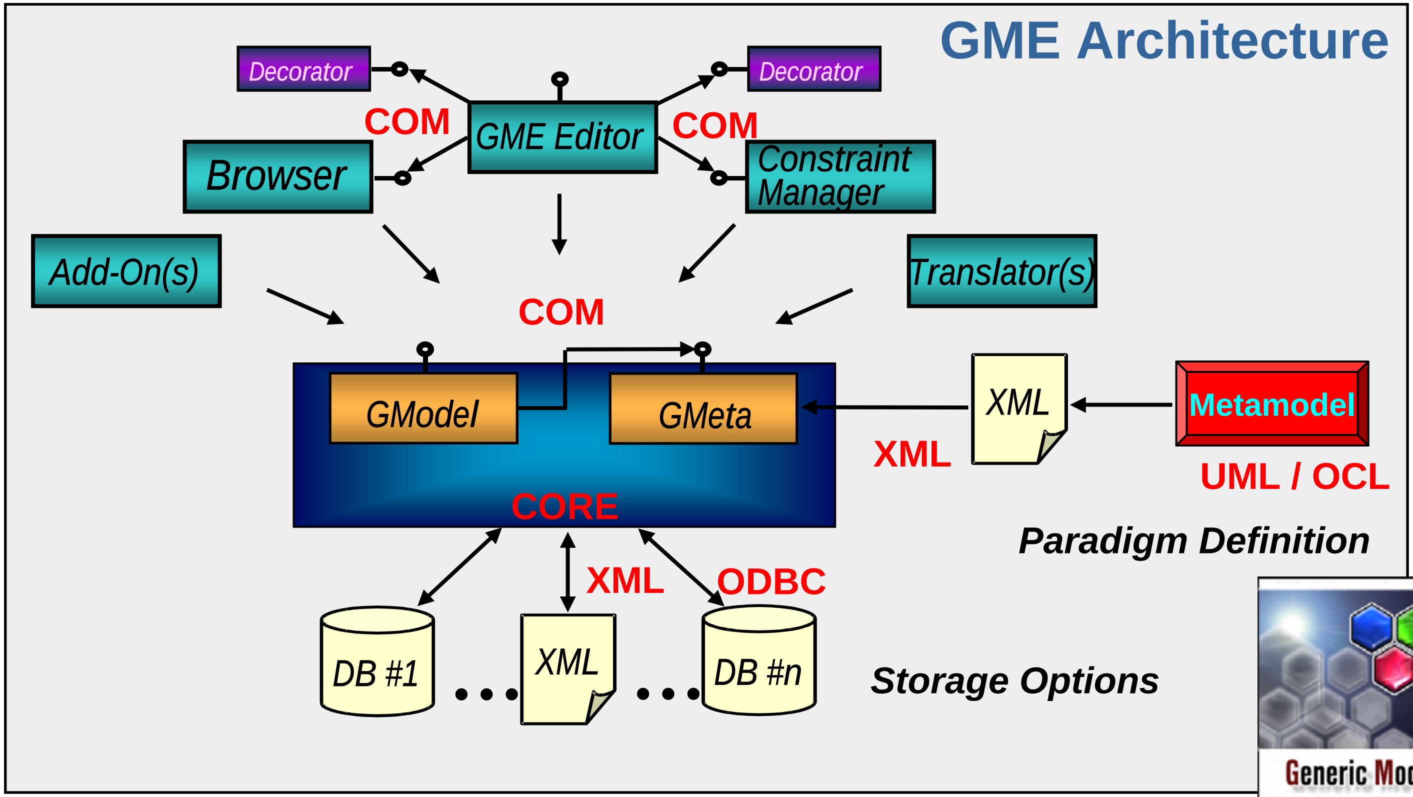
Metamodel:

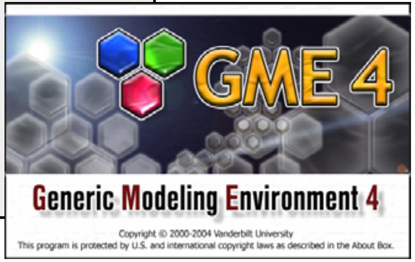
- Defines the set of admissible models
- "Metaprogramms" tool

Model:

- Describes states and transitions
- Modeling tool enforces constraints

Metaprogrammable Modeling Tool: GME



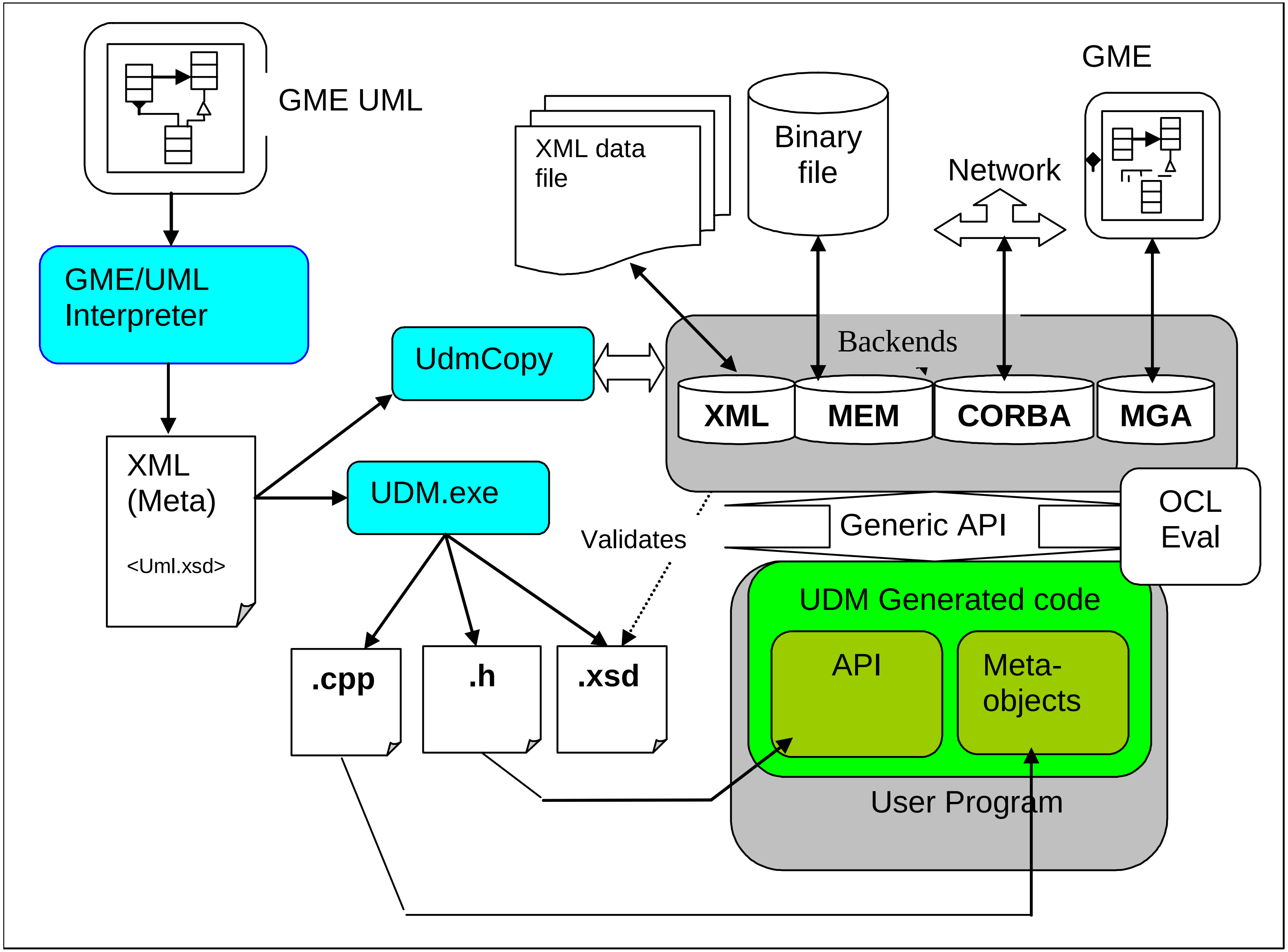


- Configuration through UML and OCL-based metamodels
- Extensible architecture through COM
- Multiple standard backend support (ODBC, XML)
- Multiple language support: C++, VB, Python, Java, C#

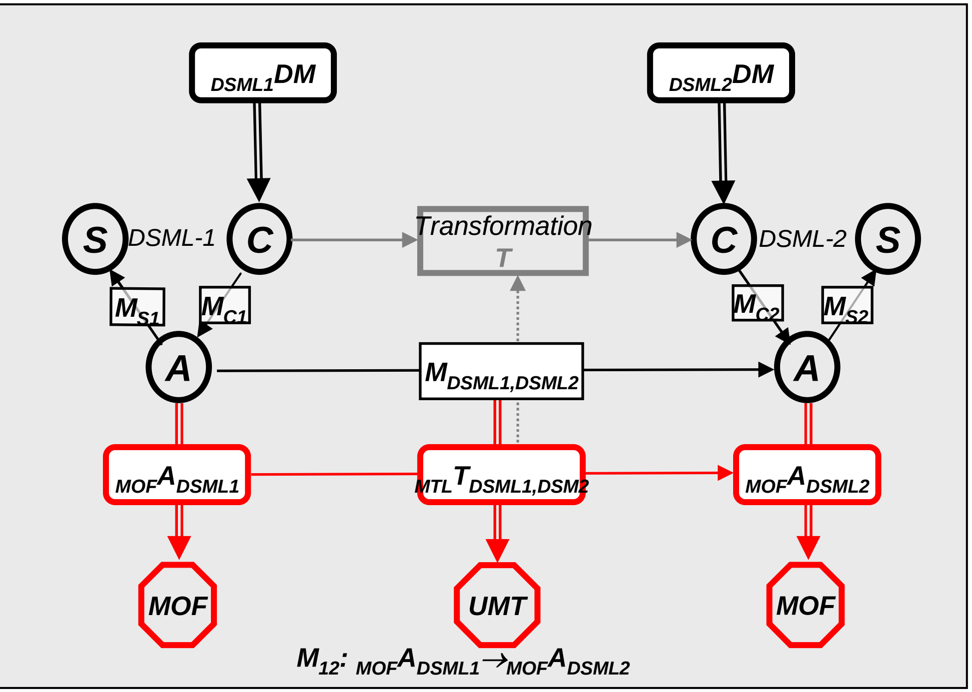
Model Data Management: The UDM Goals

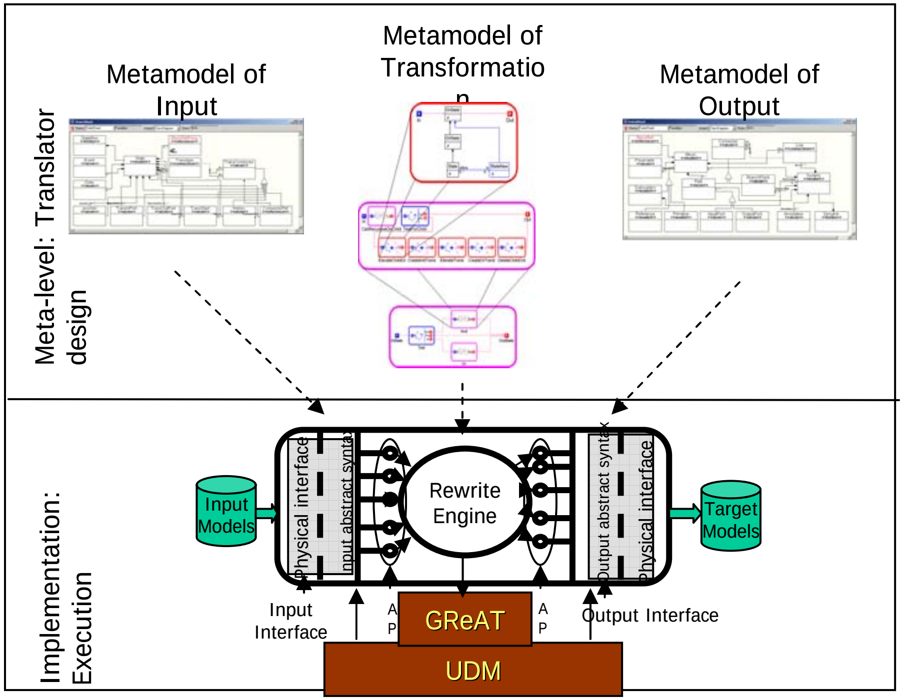
- To have a conceptual view of data/metadata that is independent of the storage format.
- Such a conceptual view should be based on standards such as UML.
- Have uniform access to data/metadata such that storage formats can be changed seamlessly at either design time or run time.
- Generate a metadata/paradigm specific API to access a particular class of data.

Model Data Management:
The UDM Tool Suite



Model Transformation: The "Workhorse" of MIC





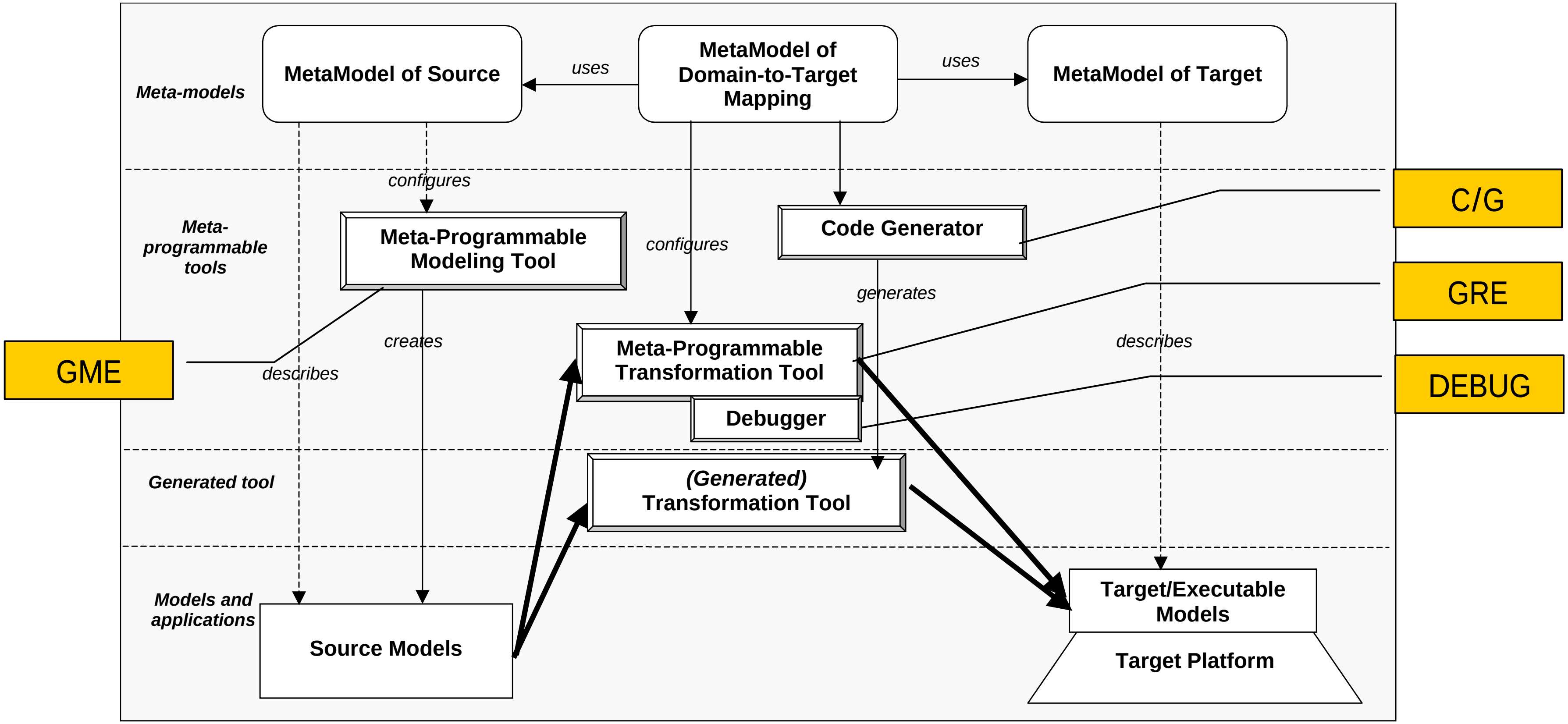
Relevant Use of Model Transformations:

- Building integrated models by extracting information from separate model databases
- Generating models for simulation and analysis tools
- Defining semantics for DSML-s

MIC Model transformation technology is:

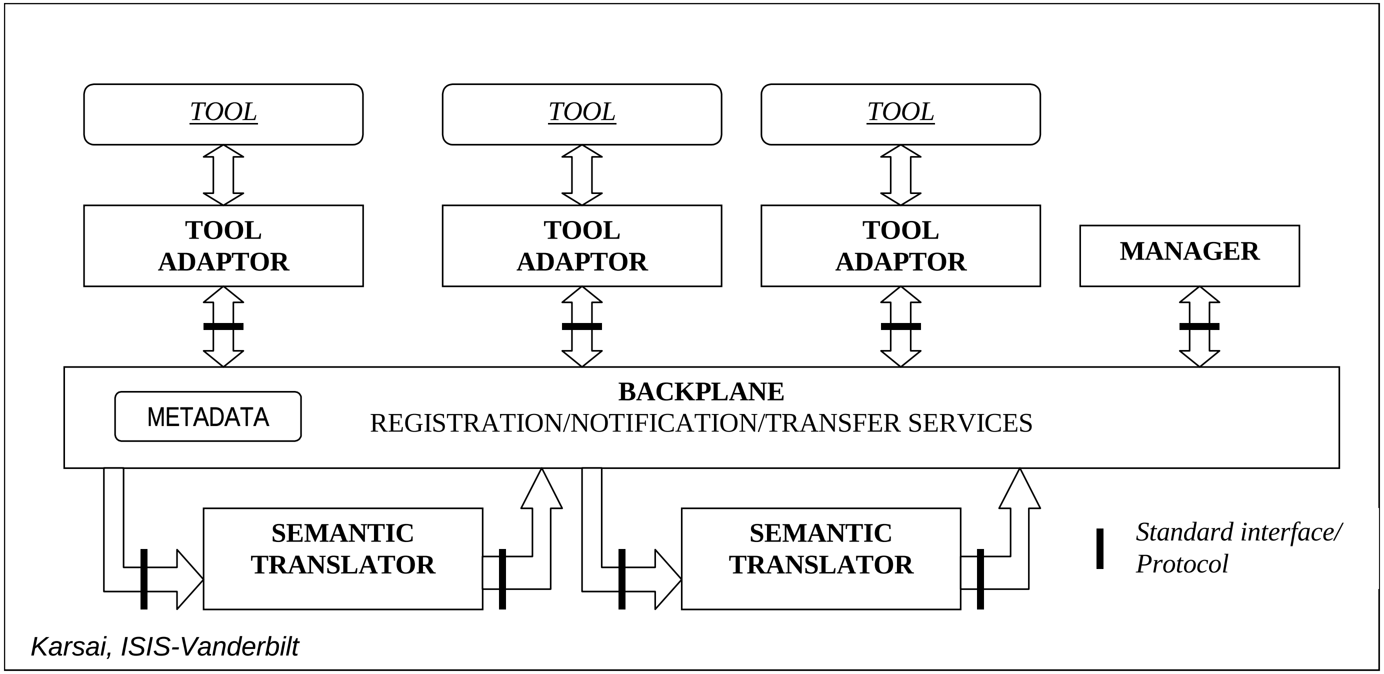
- Based on graph transformation semantics
- Model transformations are specified using metamodels and the code is automatically generated from the models.

Model Transformation: The GReAT Tool Suite



Tools: UMT Language, GRE (engine), C/G, GR-DEBUG

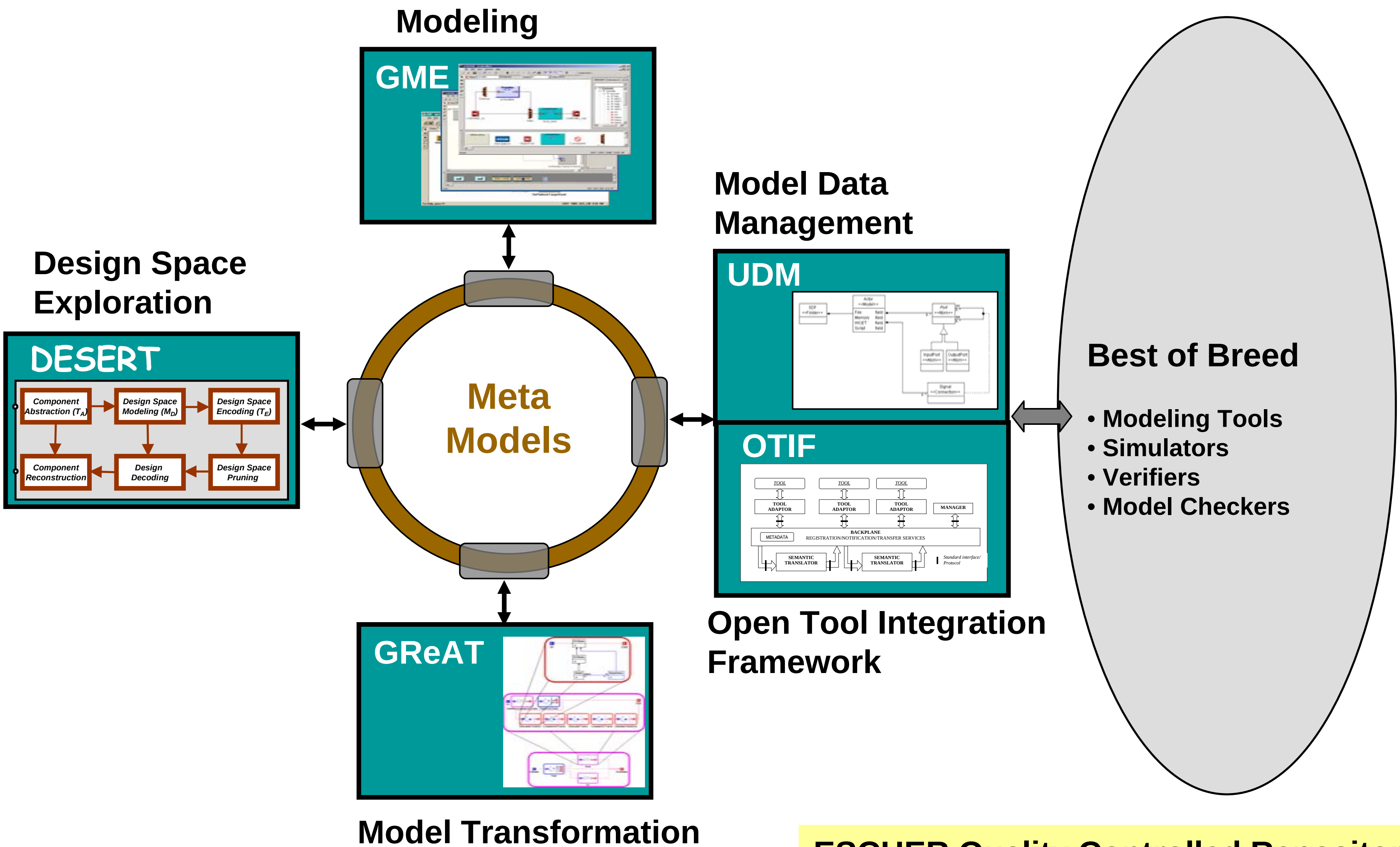
Open Tool Integration Framework: OTIF



RFP is Discussed at
MIC PSIG
OMG

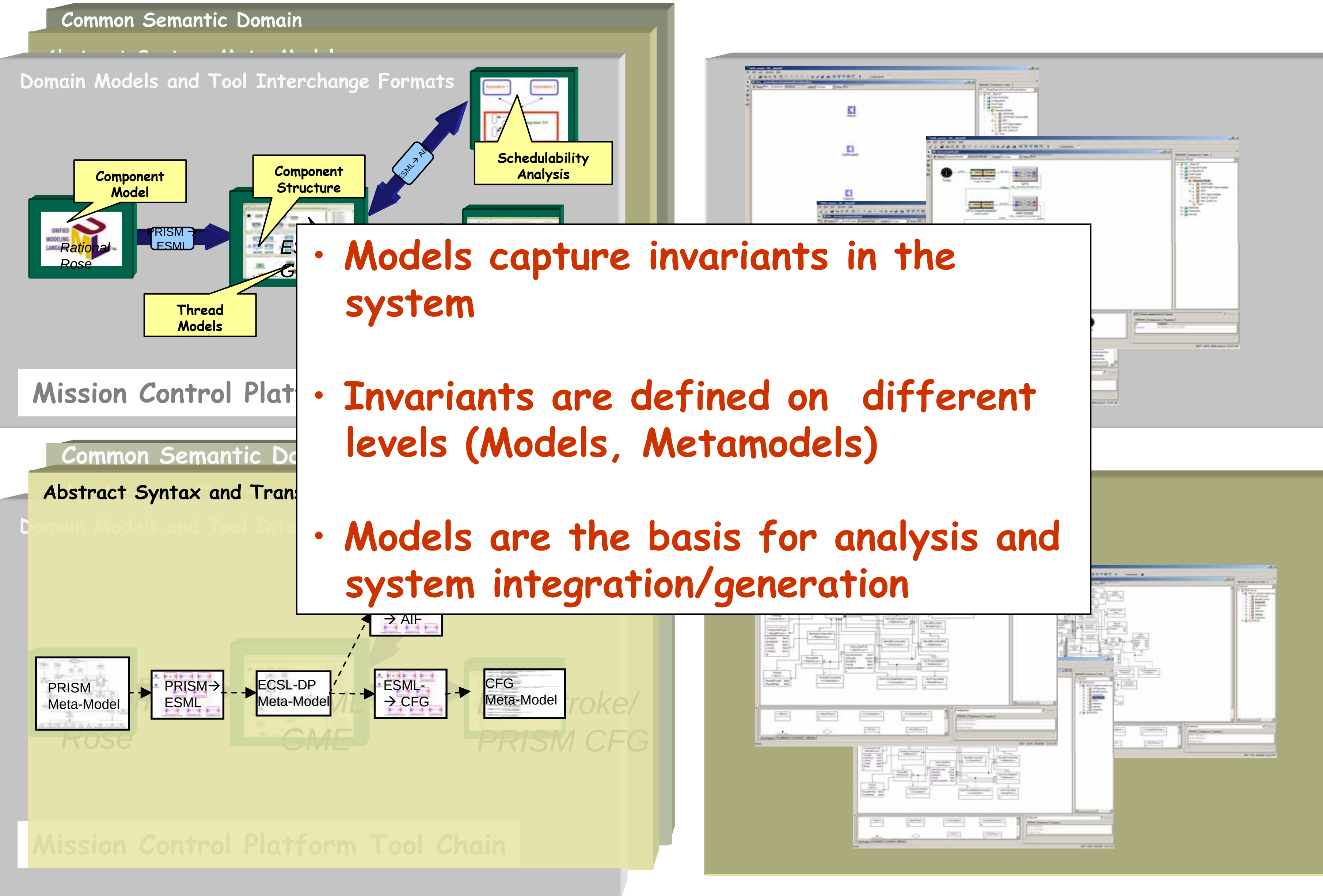
- **Share models** using Publish/Subscribe Metaphor
- Status:
  - Completed, tested in several tool chains
  - Protocols in OMG/CORBA
  - CORBA as a transport layer
  - Integration with ECLIPSE is in progress

Integrated MIC Tool Suite



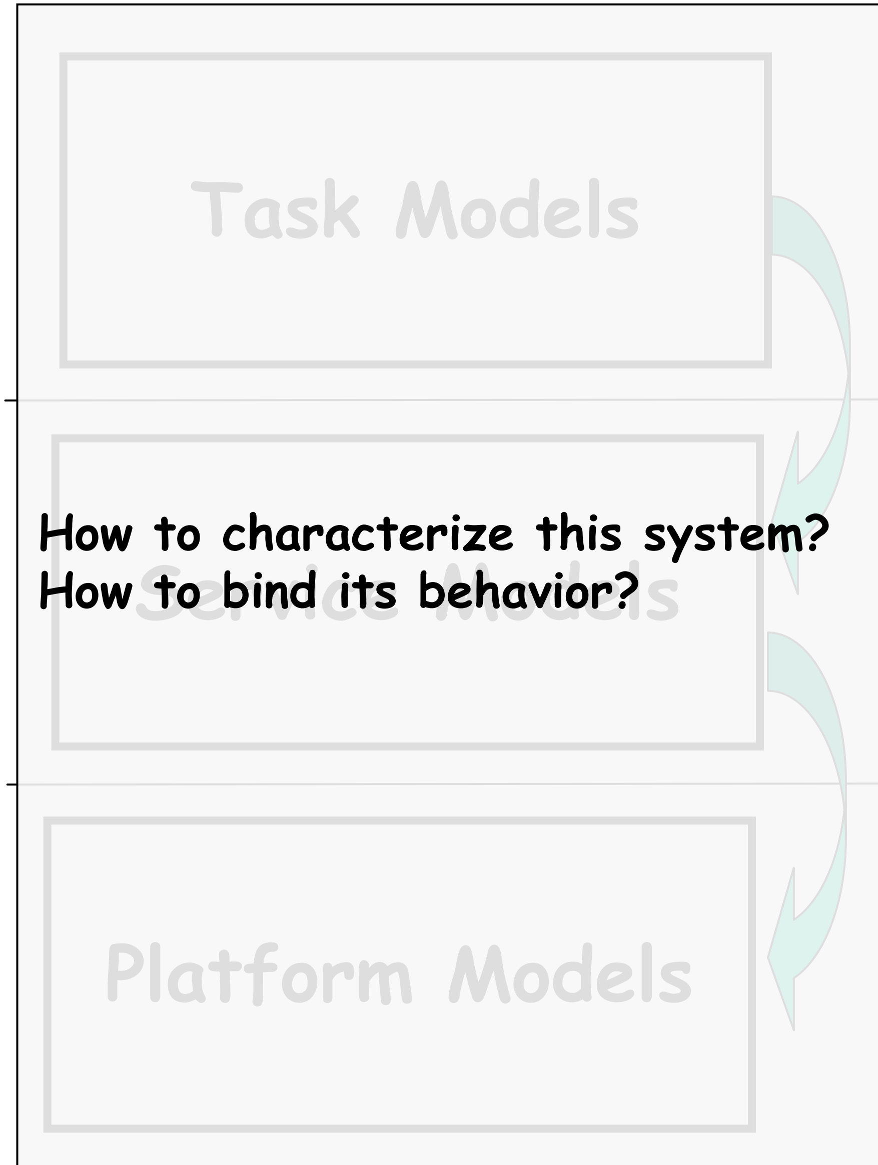
ESCHER Quality Controlled Repository:
http://escher.isis.vanderbilt.edu

Static Architectures



- Models capture invariants in the system
- Invariants are defined on different levels (Models, Metamodels)
- Models are the basis for analysis and system integration/generation

Dynamic Architectures I: Service Models



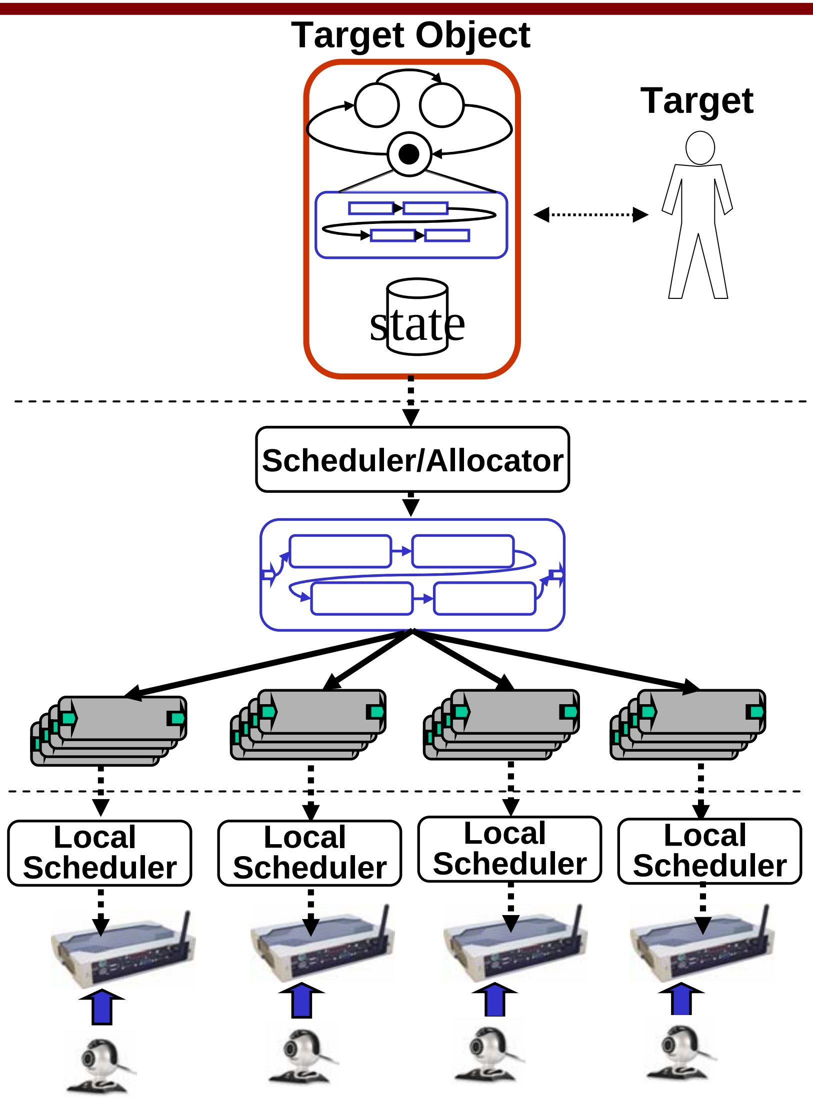
- Heterogeneous MoC-s
- Describe mode-dependent service use
- Dynamic, data-driven instantiation of instances
- Migration across platform nodes

- Heterogeneous MoC-s
- Dynamic binding to Task Model instances
- Platform-dependent instantiation and replication of services

- Distributed
- Dynamic, changing interconnections
- Error-prone communication
- Changing configurations

Service-Oriented Architecture for Sensor Networks

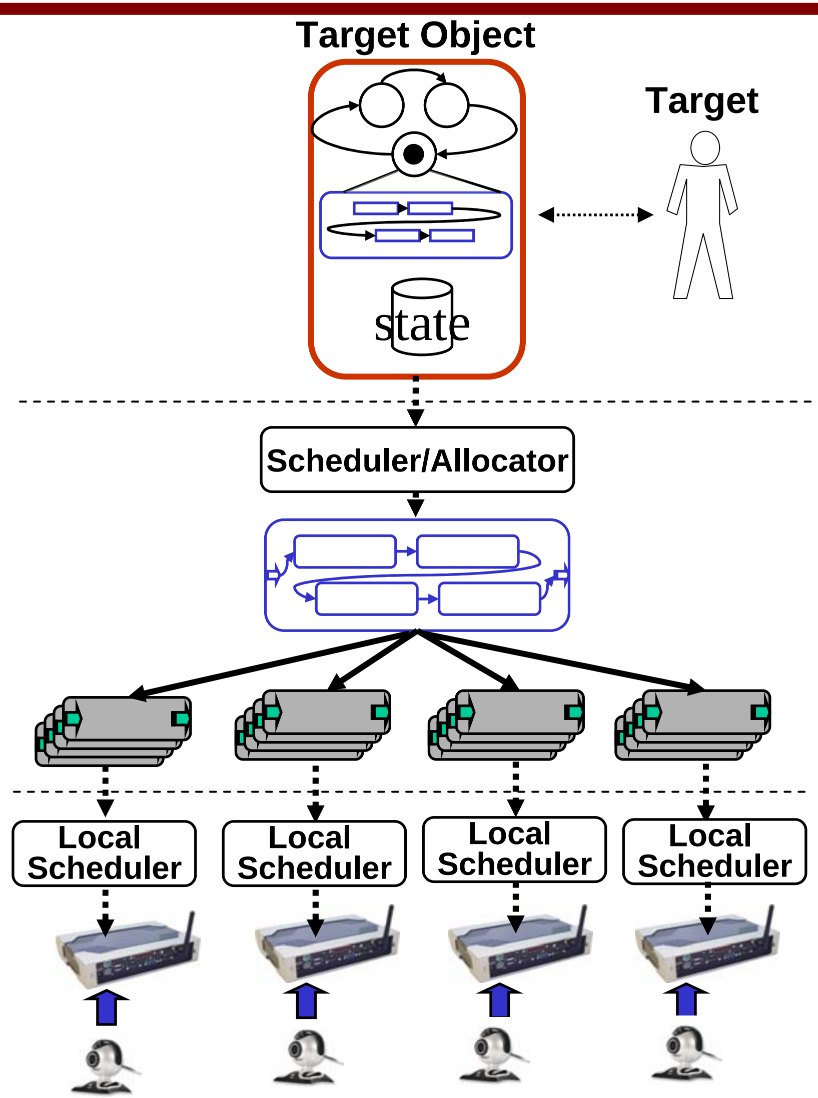
- Target is an entity (e.g. human) being sensed and serviced
- Target Object: Representation of the target that drives the application
  - A finite state machine with different modes
  - A task graph for each mode capturing the desired processing
  - The state of the target (distinct from modes) including location, motion, etc.
- A Target Object can
  - Migrate from one sensor node to another
  - Switch modes based on sensor information



In collaboration with Xenofon Koutsoukos, Vanderbilt and Wayne Wolf, Princeton

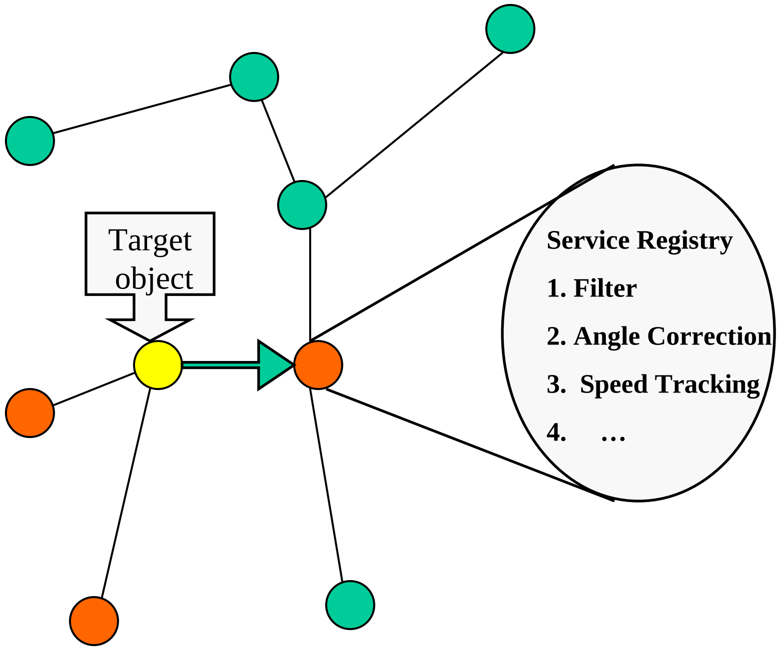
Application and Middleware Services

- Scheduler/Allocator
  - A run-time system dynamically binds tasks to application services
  - Using an application service requires only its name and interfaces
- The scheduler/allocator employs middleware services
  - Distributed discovery service
  - Binding service
  - Data distribution service



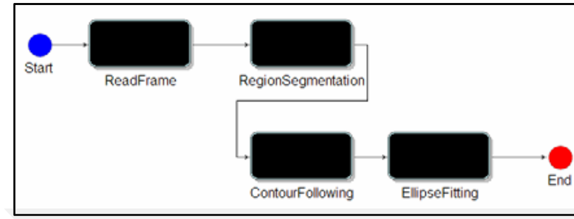
Scheduler/Allocator

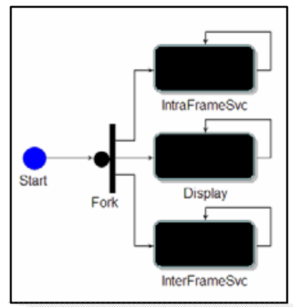
- Distributed discovery service
  - Query neighboring nodes' service registry
  - Create a local model of available services
- Binding service
  - Local operation space exploration using constraint satisfaction
- Local scheduling of services at each node

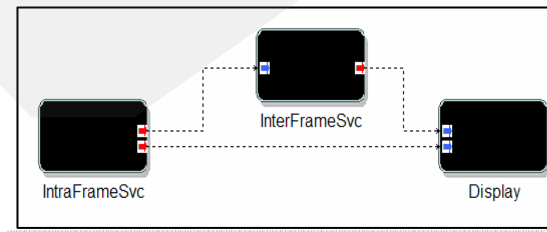


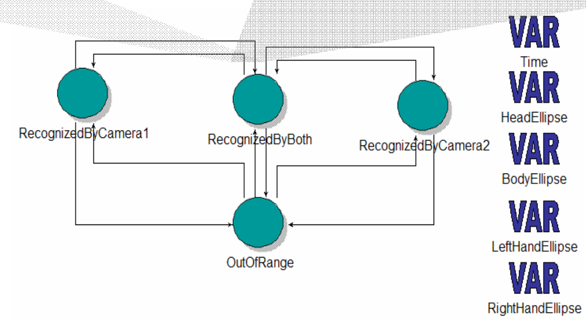
Use of MIC Tools and Methods

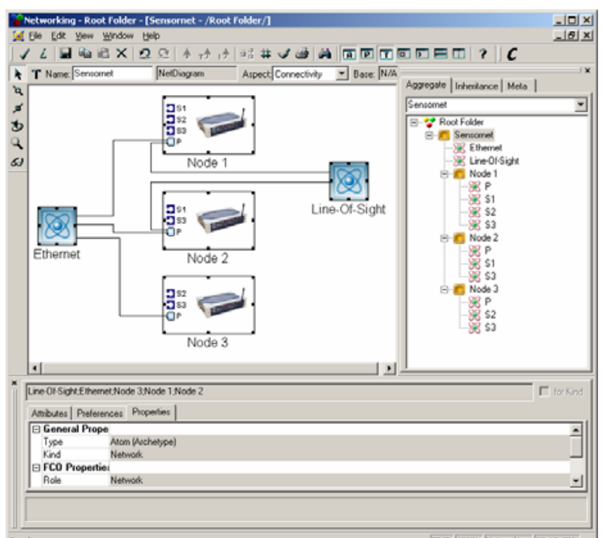
- Modeling languages for
  - Modes
  - Task flows
  - Service composition
  - System architecture
  - Data







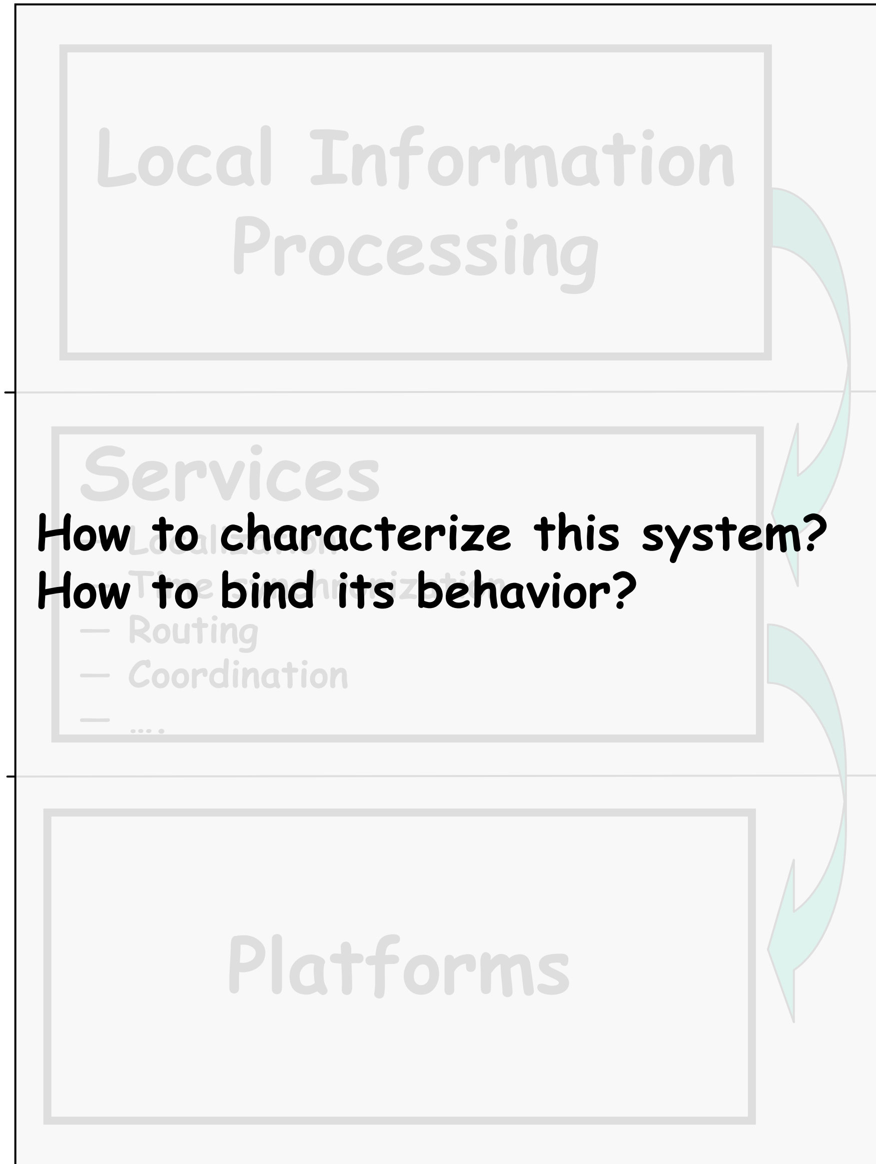




DDS: Emerging Standard for Real-Time Data Distribution

- **Data-centric** interaction and **coordination** of activities
- Distributed data space; interaction through "topics"
- **Dynamic interaction patterns (Publish/Subscribe)**
- The system **continuously changing**
- Mixed (soft, hard, or quasi) **real-time** interaction requirements
  - Primary concern: ***efficient data distribution*** with minimal overhead
  - Requires ability to **control QoS properties**: predictability, overhead, and resources used
  - *Scaling is a critical issue*
- Reliability and fault tolerance is required
- http://www.omg.org/technology/documents/formal/data_distribution.htm

Dynamic Architectures II: Sensor Networks



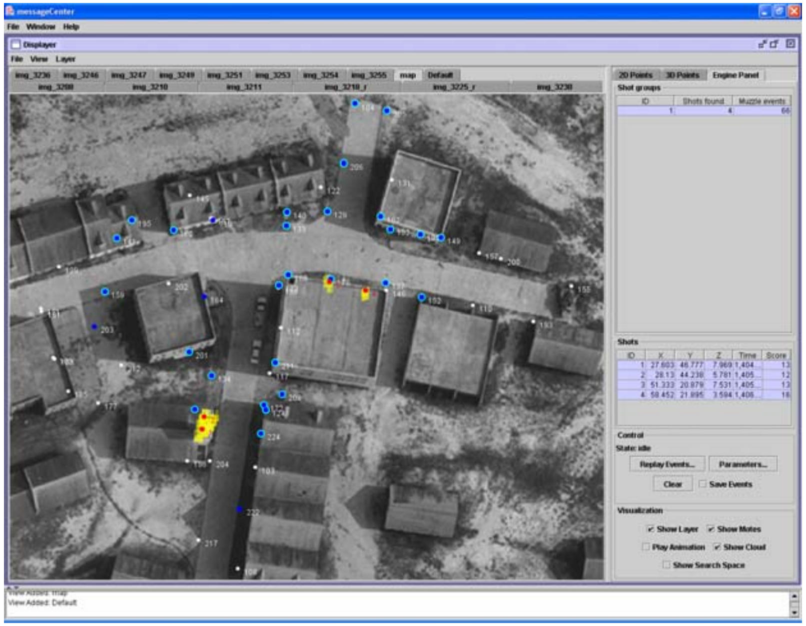
How to characterize this system?
How to bind its behavior?

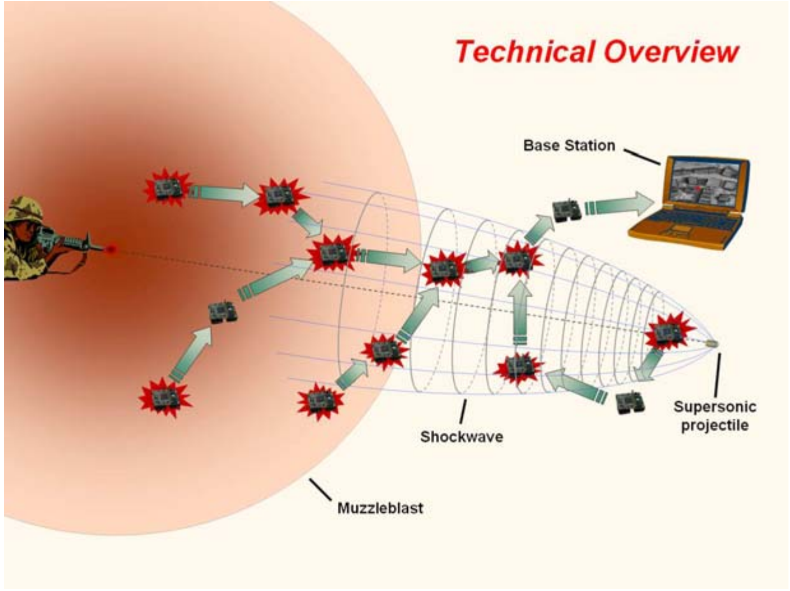
- Heterogeneous MoC-s
- Describe real-time sensor processing
- Power aware algorithms
- Produces compressed data for migrating across platforms (possibly to base station)

- Complex state automata
- Tight dependence onbthephysical properties of the platform
- “Emerging” global behavior

- Fine-grain distributed
- Dynamic, highly uncertain interconnections
- Error-prone computation nodes
- Continuously changing configurations

Example: Vanderbilt Shooter Localization System





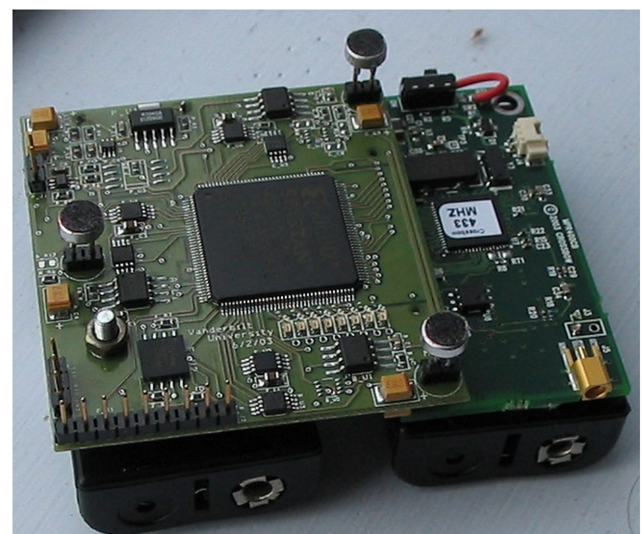
- → Urban environment with echo and no line-of-site
- → Rapid deployment and low cost
- → Multiple simultaneous shot resolution
- → Idea: Sensor network with cheap acoustic sensors, exploiting redundancy
- → Challenges:
  - Severely resource constrained nodes
  - Very limited communication bandwidth
  - Significant multipath effects in urban environment

- → Solution developed by an ISIS team between 2003-2005
  (Maroti M., Simon G., Ledeczi A., Sztipanovits J.: **Shooter Localization in Urban Terrain**, Computer, 37(8), 60-61, 2004.)

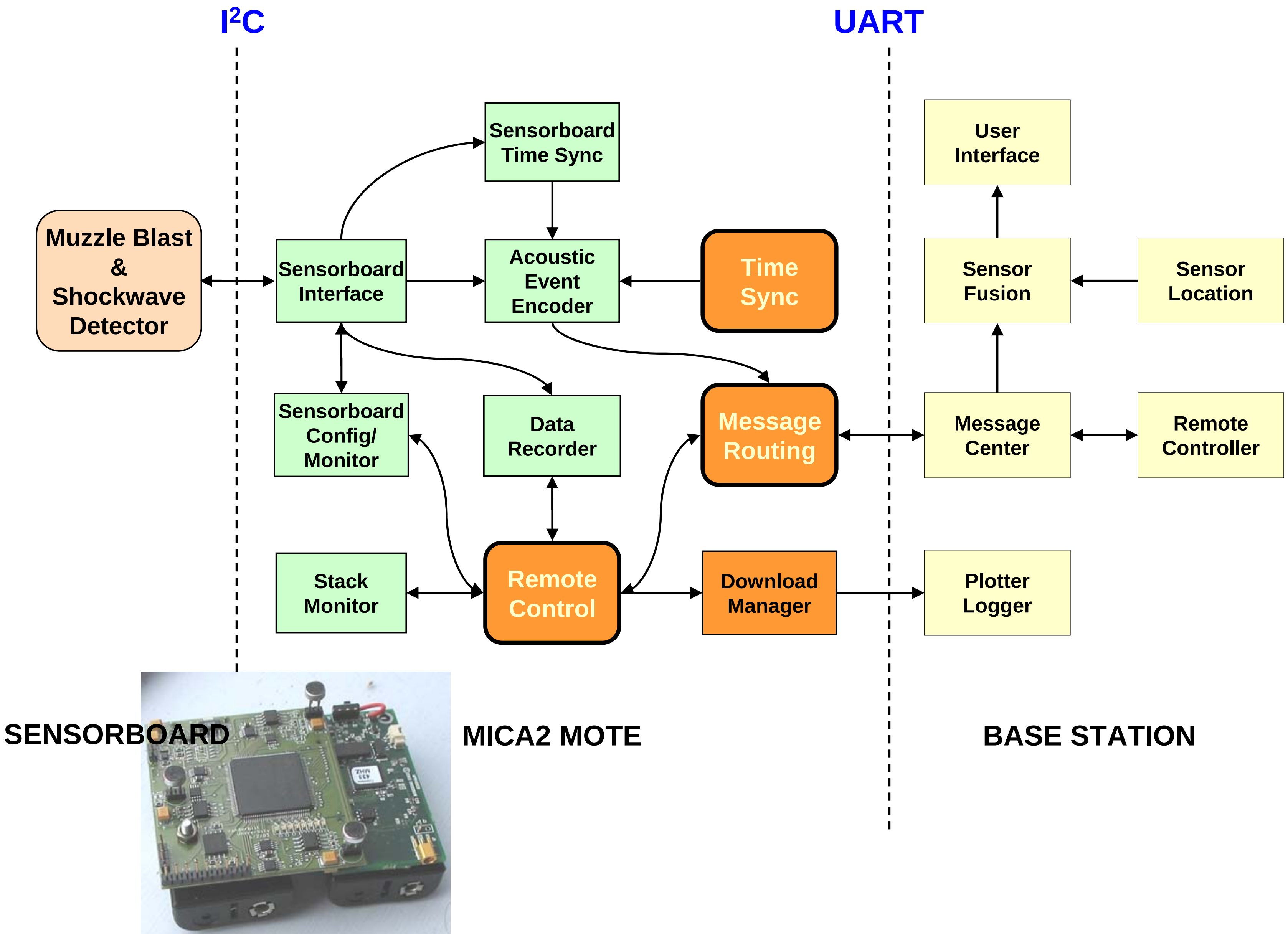
Technical Approach

- **Detect Time of Arrival (TOA) of acoustic shockwave and muzzle blast**
  - Application specific acoustic sensor board:
    - 3 acoustic channels (only a single channel is used in final system)
    - High-speed AD converters
    - FPGA for signal processing: shockwave and muzzle blast detection on board
- **Timestamp of shockwave and/or muzzle blast sent to Mica2 mote**
- **Mica2 motes route TOA data to base station**
- **Base station fuses data, estimates shooter position and displays result**
- **Middleware services:**
  - *Localization*
  - *Time synchronization*
  - *Message routing*
  - *Remote control*
- **Tiny OS operating system ad-hoc networking**

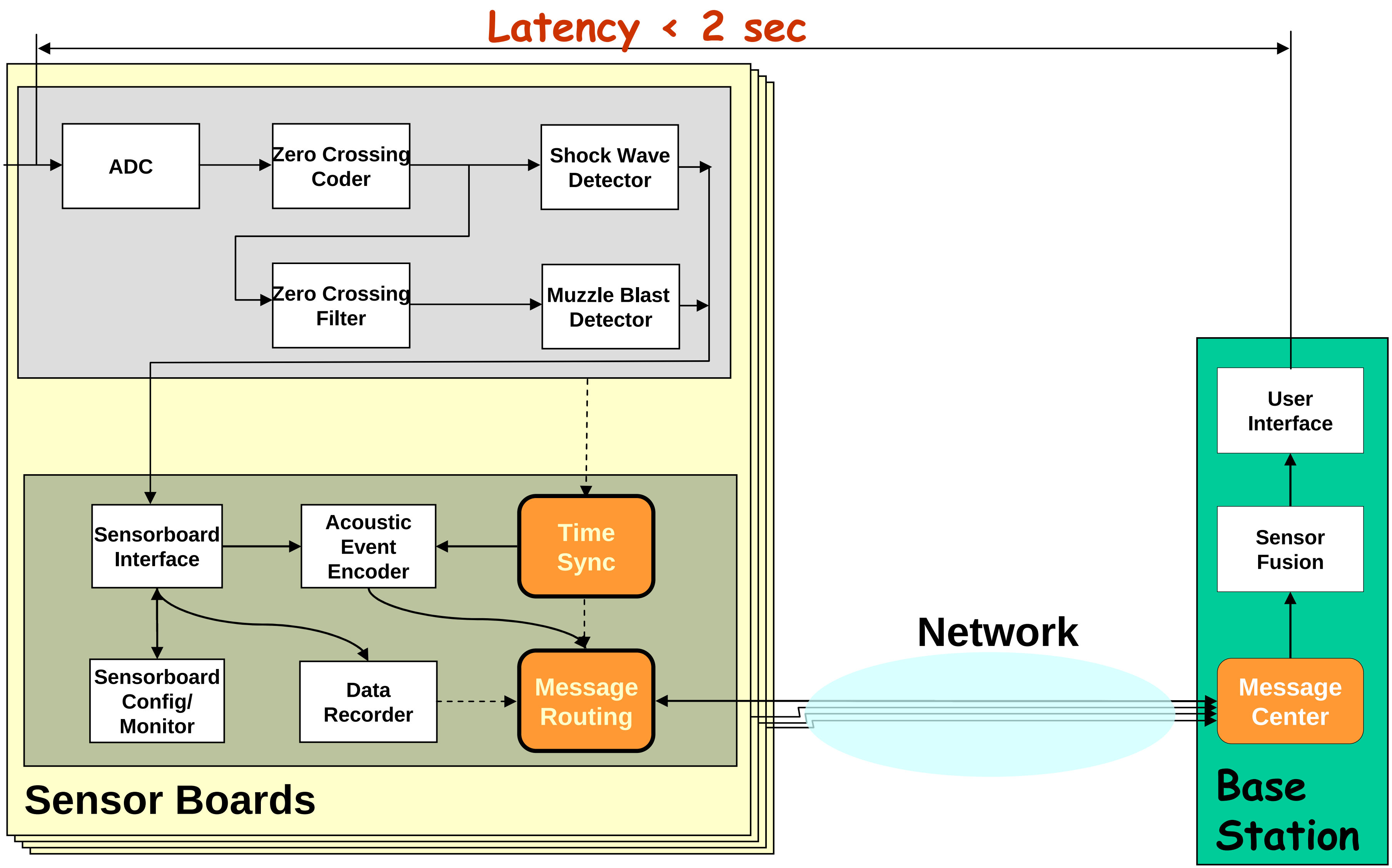
  (Ledeczi et.al."Countersniper System for Urban Warfare", ACM TOSN, 2(1), 153-177, 2005.)



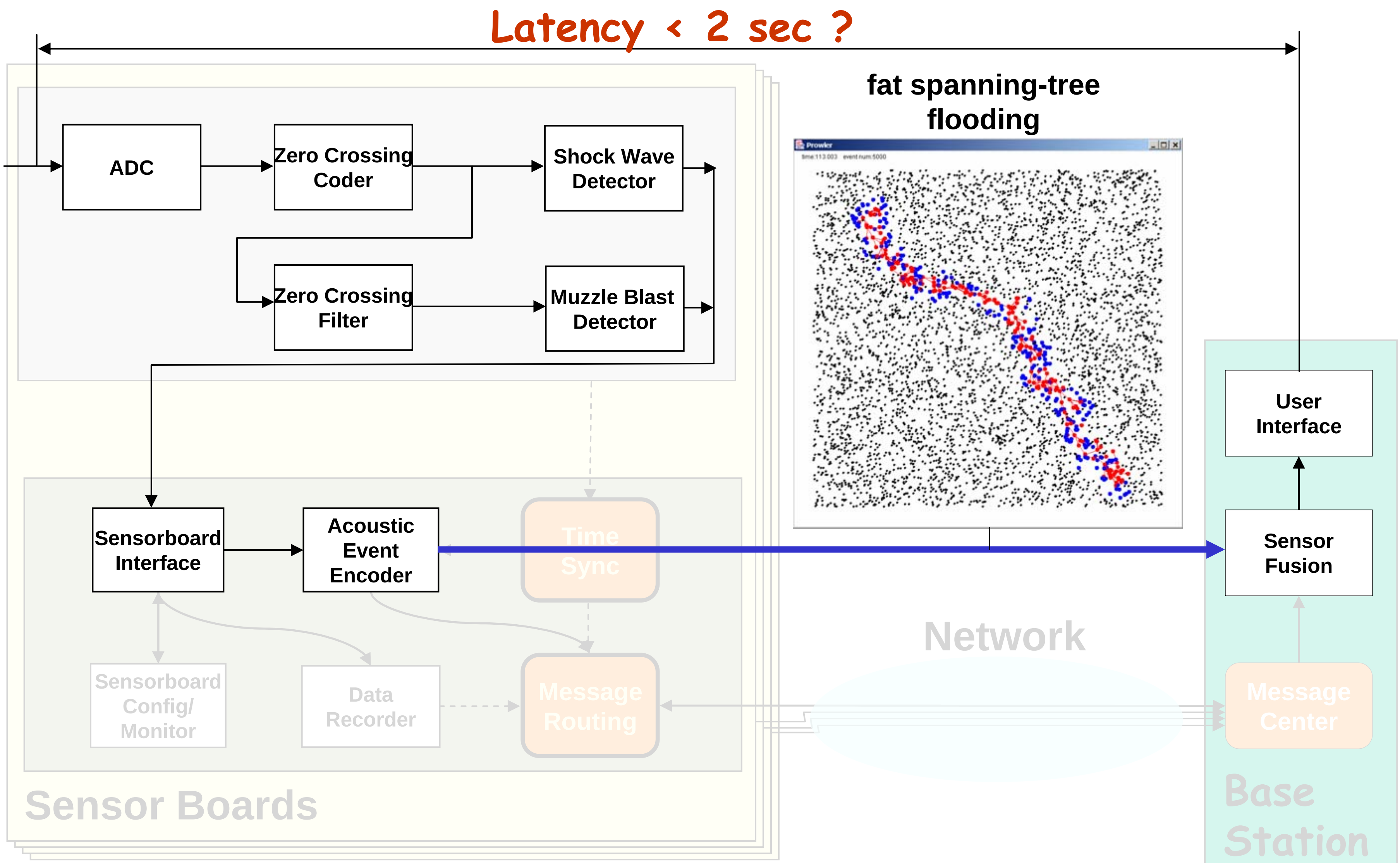
System Architecture



Unique Challenges: Latency

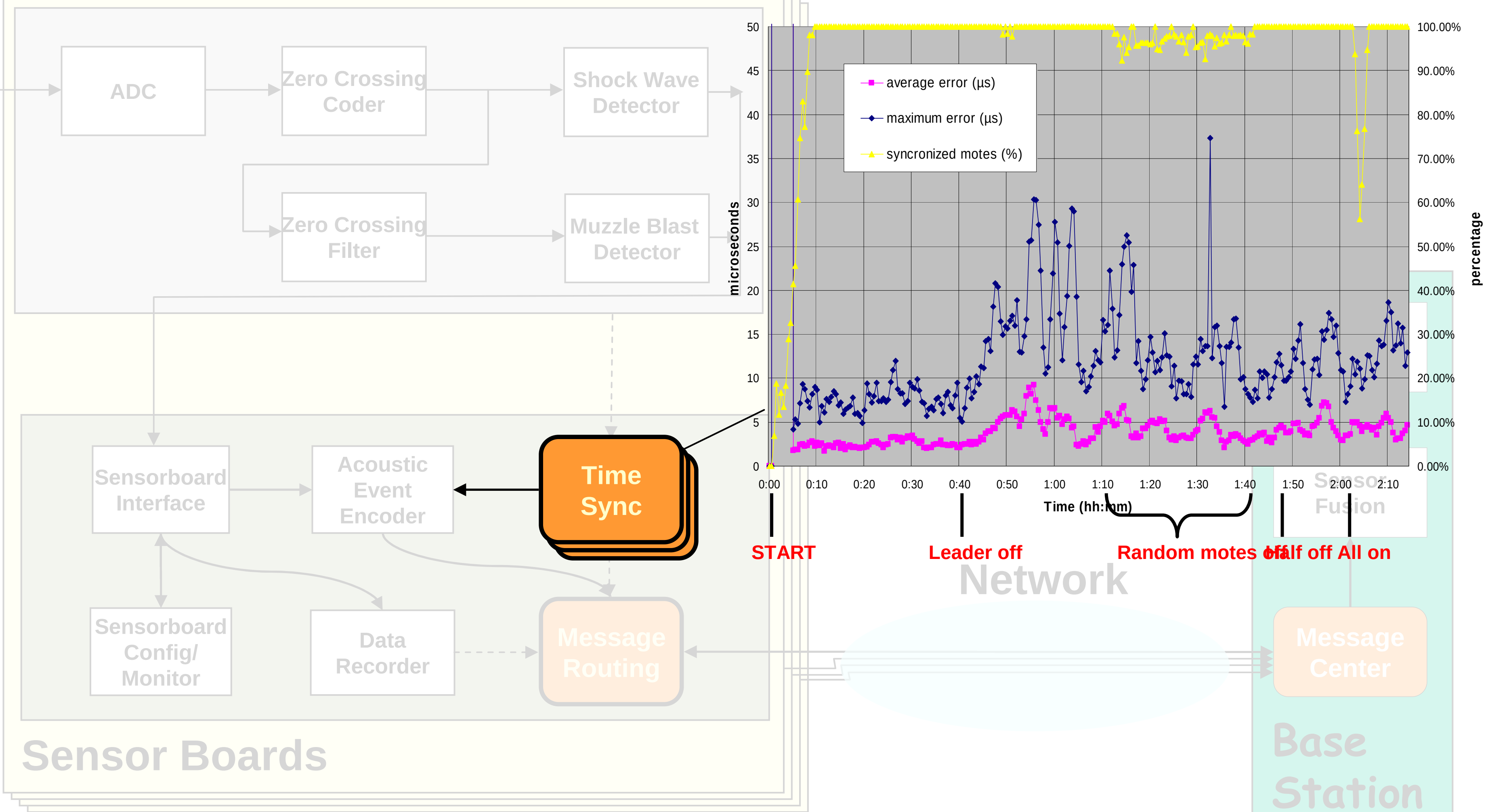


Unique Challenges: Latency

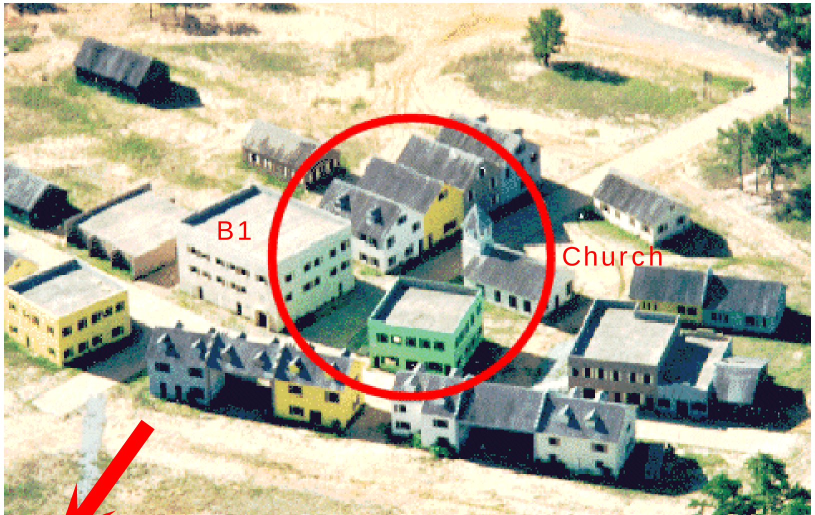


Unique Challenges: Time Synch

Flooding Time Synchronization Protocol (FTSP)



Real-life Experiments



NORTH

- Sep 2003: Baseline system
- Apr 2004: Multishot resolution

- 60 motes covered a 100x40m area
- Network diameter: ~7 hops
- Used blanks and Short Range Training Ammunition (SRTA)
- Hundreds of shots fired from ~40 different locations
- Single shooter, operating in semiautomatic and burst mode in 2003
- Up to four shooters and up to 10 shots per second in 2004
- Variety of shooter locations (bell tower, inside buildings/windows, behind mailbox, behind car, …) chosen to absorb acoustic energy, have limited line of sight on sensor networks
- Hand placed motes on surveyed points (sensor localization accuracy: ~ 0.3m)

Conclusions

- Network Centric Systems offer completely new solutions for old, very hard problems
- Model-based design and tools are indispensable in their design.
- Application design frequently spans DSP/HW/SW/Networking with complex interdependences
- Modeling paradigms are more complex, heterogeneous and model integration is becoming a major challenge