DEVS Standardization

Joint Session –

TMS/DEVS - SIMAUD

Boston, MA - April 5, 2011

Current practices in M&S

- Evolution in M&S: driven by technology (computing power, networks, graphical interfaces, standards).
- Current practices: **ad-hoc** techniques tailored to solve each particular problem.
- Tendency to **encapsulate** models, simulators and experiments into **tightly coupled** packages (written in programming languages such as Fortran, C, C++, Java).
- **Difficulties**: testing, maintainability of the applications, integration, software reuse.
- Relatively few examples of storing **previously** developed models to be adapted for interoperability and reuse

Advantages of DEVS M&S methodology

- Research in the last 15-20 years showed that DEVS theory can is a good starting point to provide:
  - Interoperability and reuse
  - Engineering-based approach (different for varied types of M&S life cycles)
  - Facilities for automated tasks
  - High performance/distributed simulation
  - Hybrid systems definition

DEVS Toolkits

- **ADEVS** (University of Arizona)
- **CD++** (Carleton University)
- **DEVS-C++** (Kaist – Korea)
- **DEVS/HLA** (ACIMS)
- **DEVSJAVA** (ACIMS)
- **DEVSim++** (Kaist- Korea)
- **GALATEA** (USB – Venezuela)
- **GDEVS** (Aix-Marseille III, France)
- **JDEVS** (Université de Corse - France)
- **PyDEVS** (McGill)
- **PowerDEVS** (University of Rosario, Argentina)
- **SimBeams** (University of Linz – Austria)
- New efforts in China, France, Portugal, Spain.

Where to go from now

- Bridging the gap between research and practice
  - DEVS ready to take the leap
  - Critical mass of knowledgeable people
  - Large number of tools/researchers
  - Ready to go from Research to Development
- Standardization of models (DEVS and non-DEVS)
- Building libraries/user-friendly environments
- Further research required; open areas.

Goals

- Study standard representation of DEVS to support common understanding, sharing and interoperability
- Analysis potential establishment of a core for a DEVS standard.
- Relationship to other standards
- Potential creation of a proposed standardized language

DEVS Standardization activities

Focus groups

- Team 1: Building DEVS language+libraries

  => Improving learning curve

- Team 2: implementation of services to interoperate at least two existing DEVS tools

- Team 3: discussion of the contents of a DEVS kernel

Current Developments - Team 1

Building DEVS language+libraries

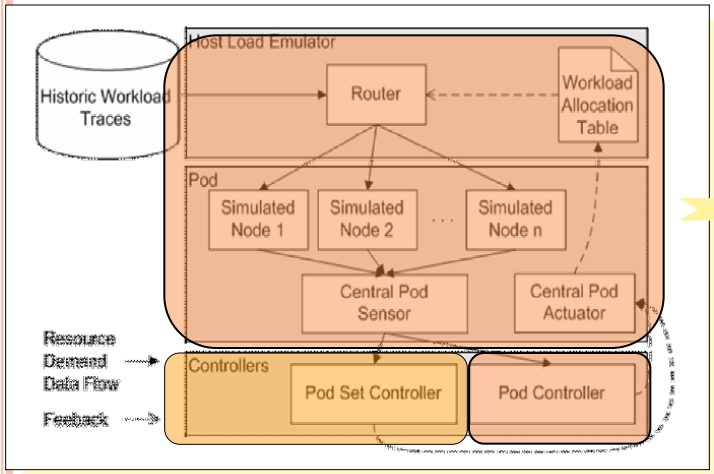
- Built versions of graphical-based tools with educational purposes
- Tools used for teaching DEVS concepts (ASU, UofA, Carleton University, McGill, Univ. Marseille, Blaise Pascal)
- Extending the experience to other members

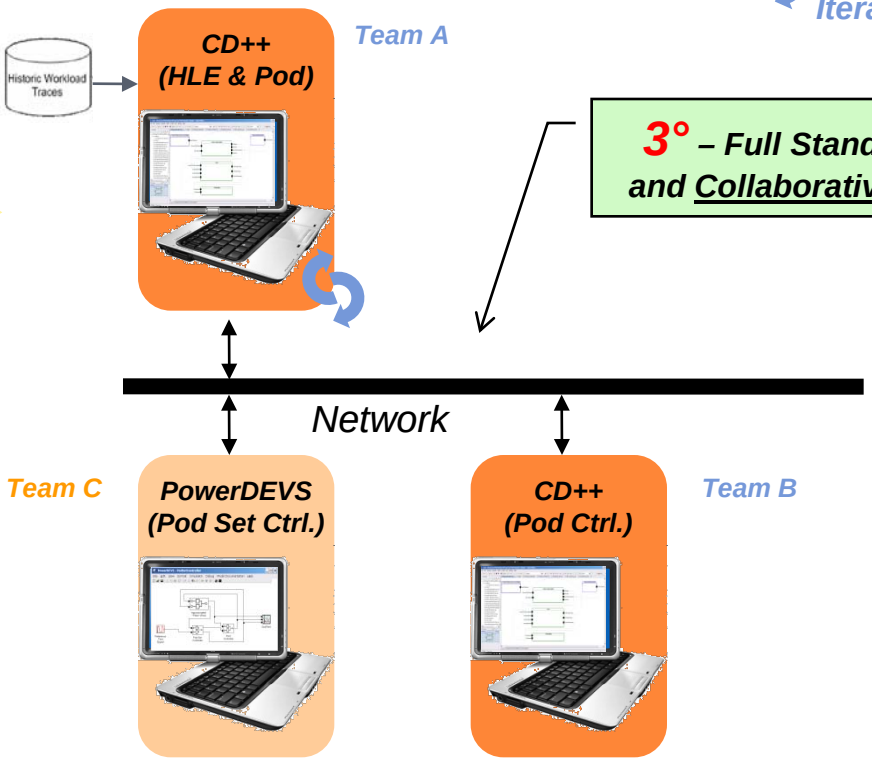
Current Developments - Team 2

Interoperation of two or more existing DEVS tools

- ADEVS and DEVSJava (H. Sarjoughian)
- CD++ and DEVS/C# (G. Wainer)
- E-CD++ and PowerDEVS (G. Wainer)
  - http://youtube.com/arslab

Introduce a new discipline/problem domain



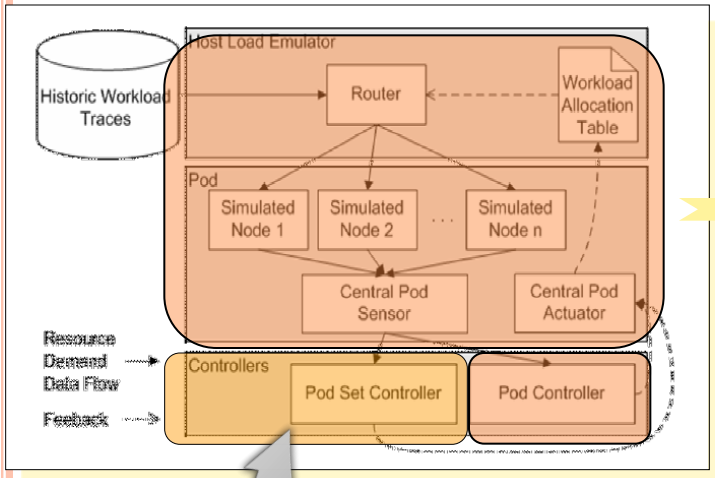


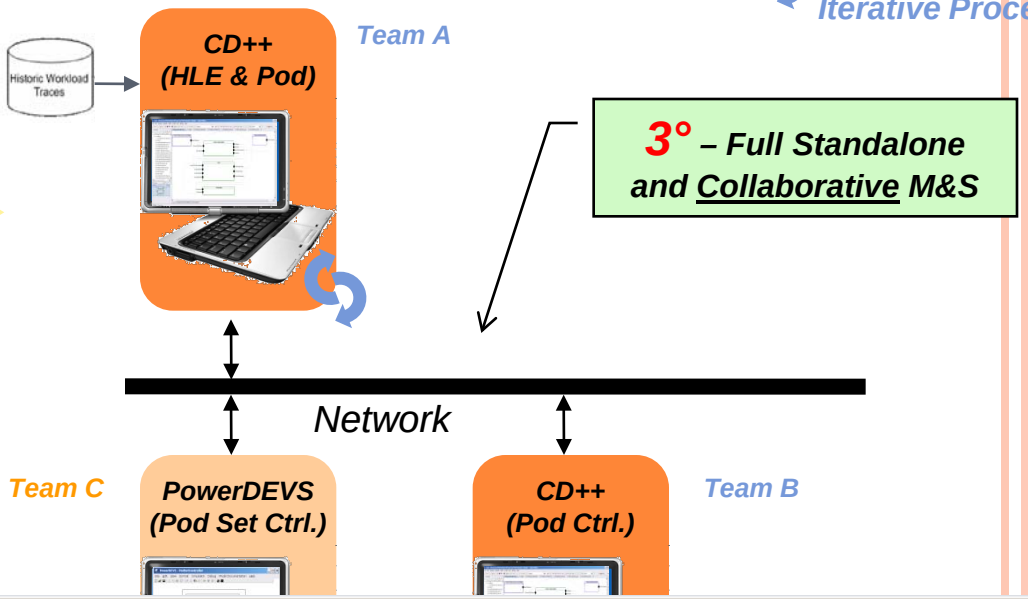
Model, Simulate, Verify and Tune Iterative Process

e.g.: Experiment with new Control Theory-based controllers interacting with the pre-existent Pod Controller.

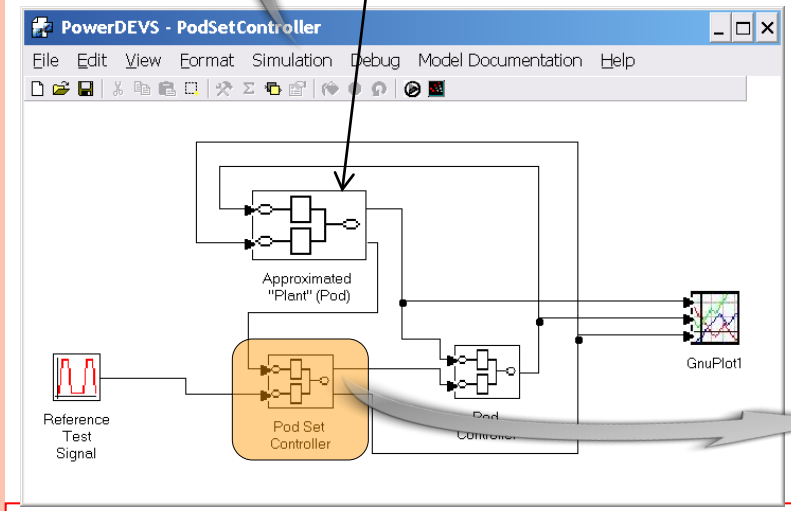
PowerDEVS for M&S of Continuous Systems

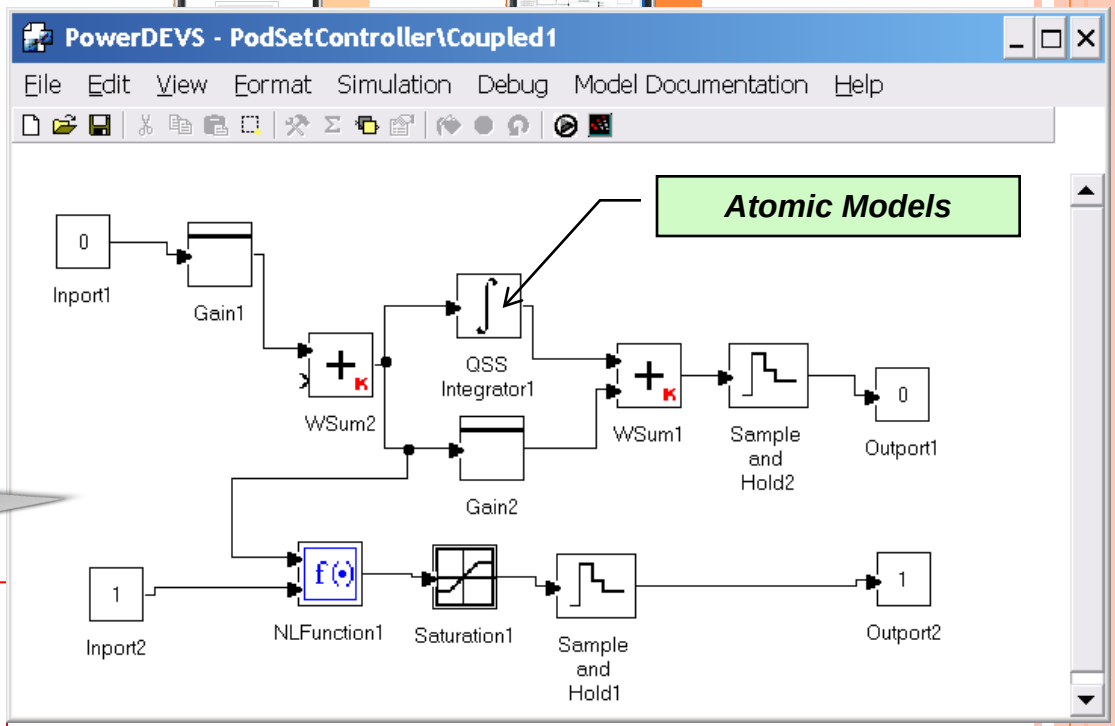
Model, Simulate, Verify and Tune Iterative Process





Coupled Models





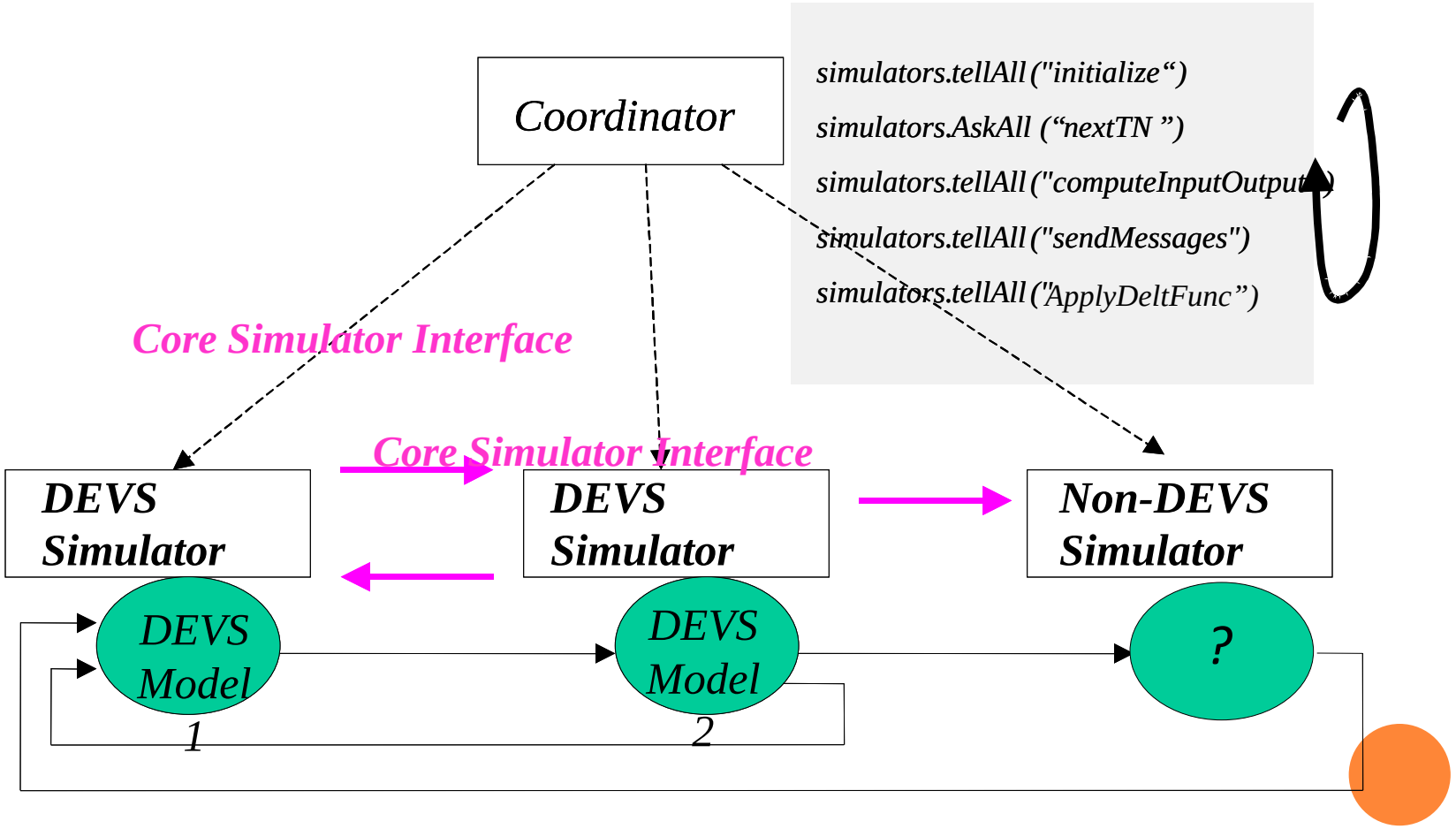
e.g.: Experiment with new Control Theory-based controllers interacting with the pre-existent Pod Controller.

Current Developments - Team 3

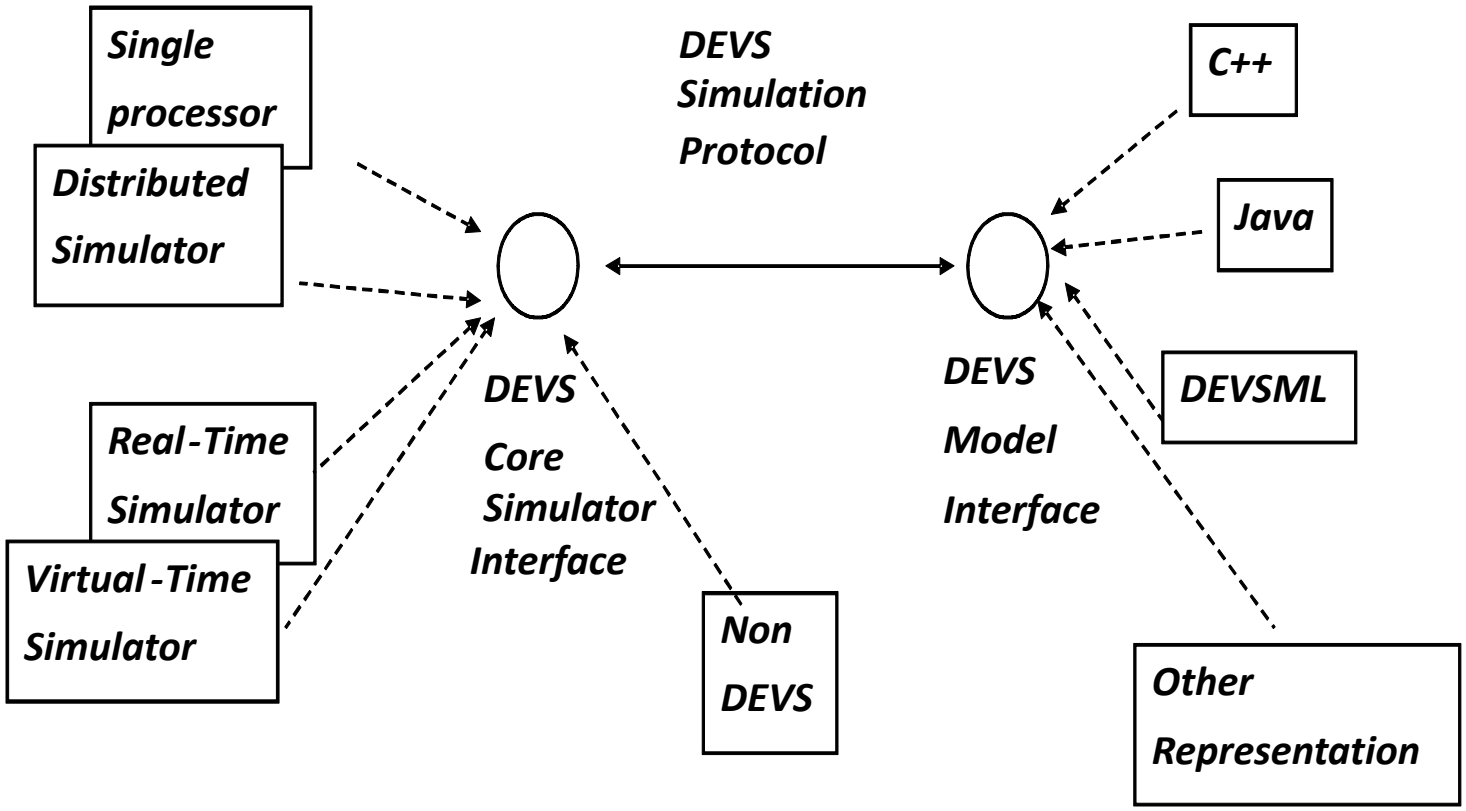
Discussion of the contents of a DEVS kernel

- Book chapter published (4)

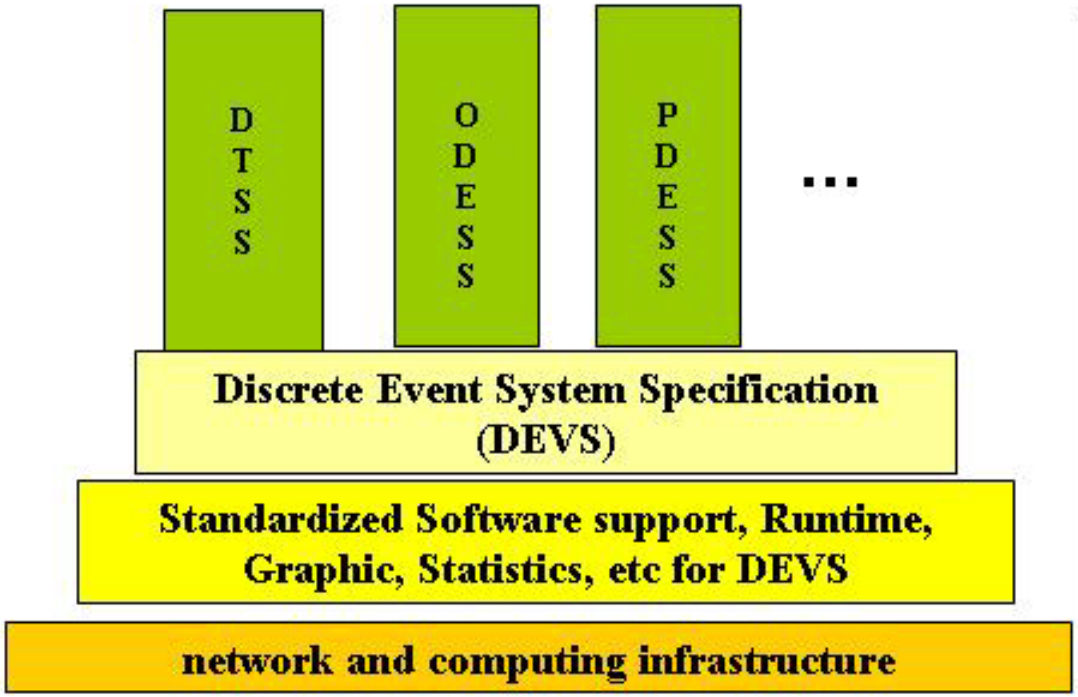
DEVS Simulation Protocol



Concept of DEVS Standard



Standardization at the right level



Objectives:

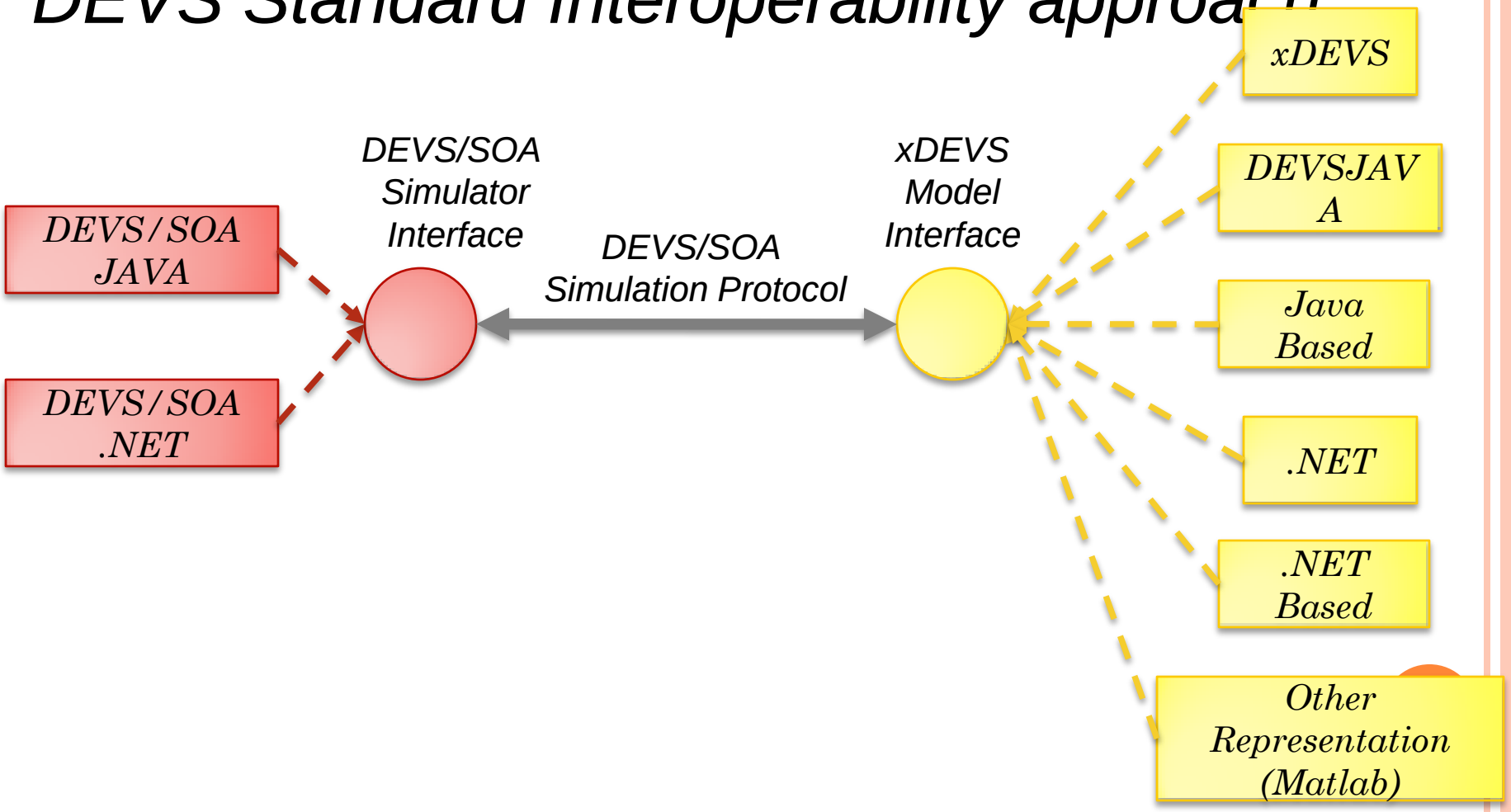
- Support interoperability at the modeling level
- Support model composability
- Technology independence
- Include all modeling paradigms in wide use
- Research collaboration academia, industry, gov

Legend

DTSS – Discrete Time System Specification
ODESS – Ordinary Differential Equation System Specification
PDESS – Partial Differential Equation System Specification

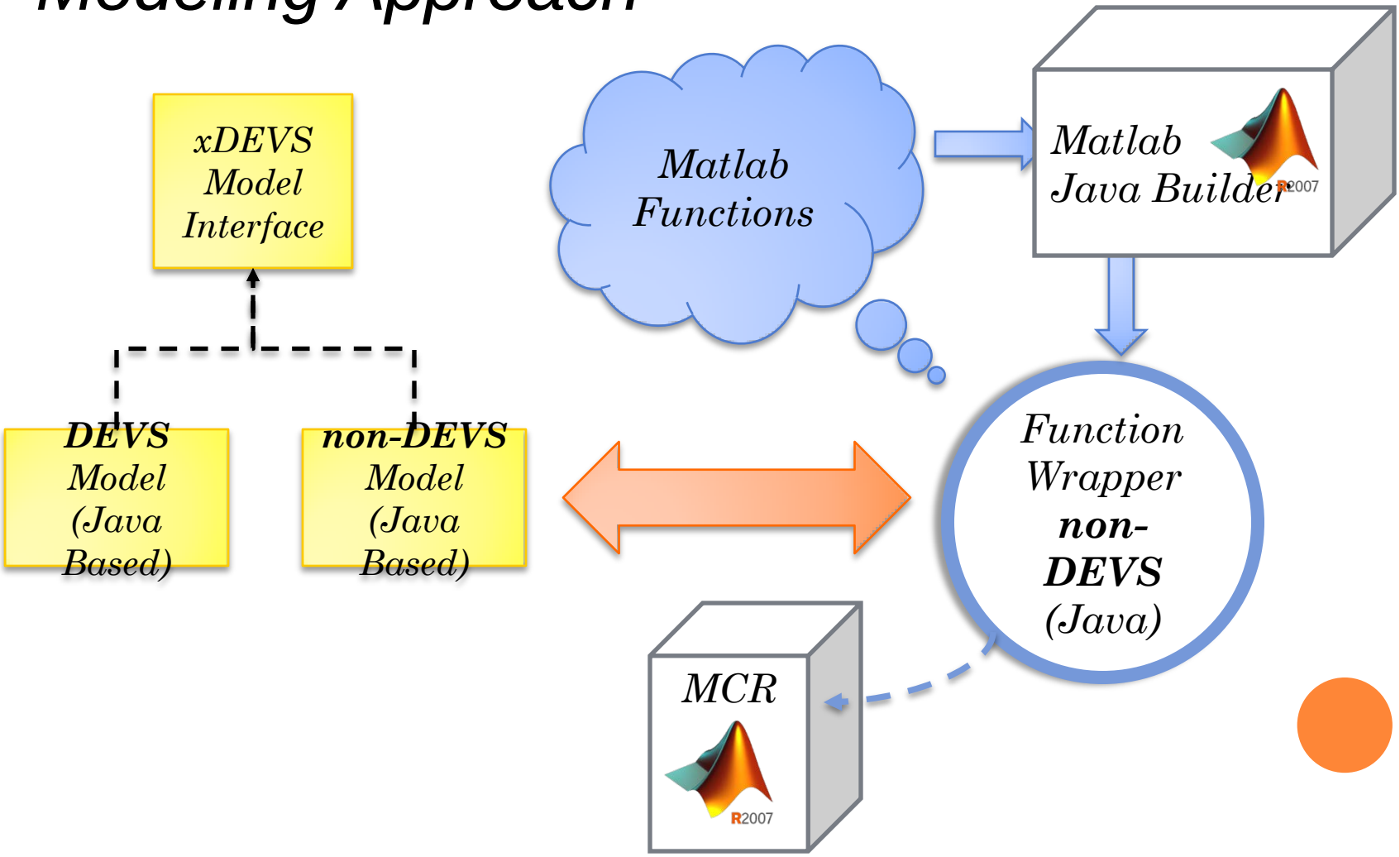
Interoperability between DEVS and non-DEVS models using DEVS/SOA

DEVS Standard Interoperability approach



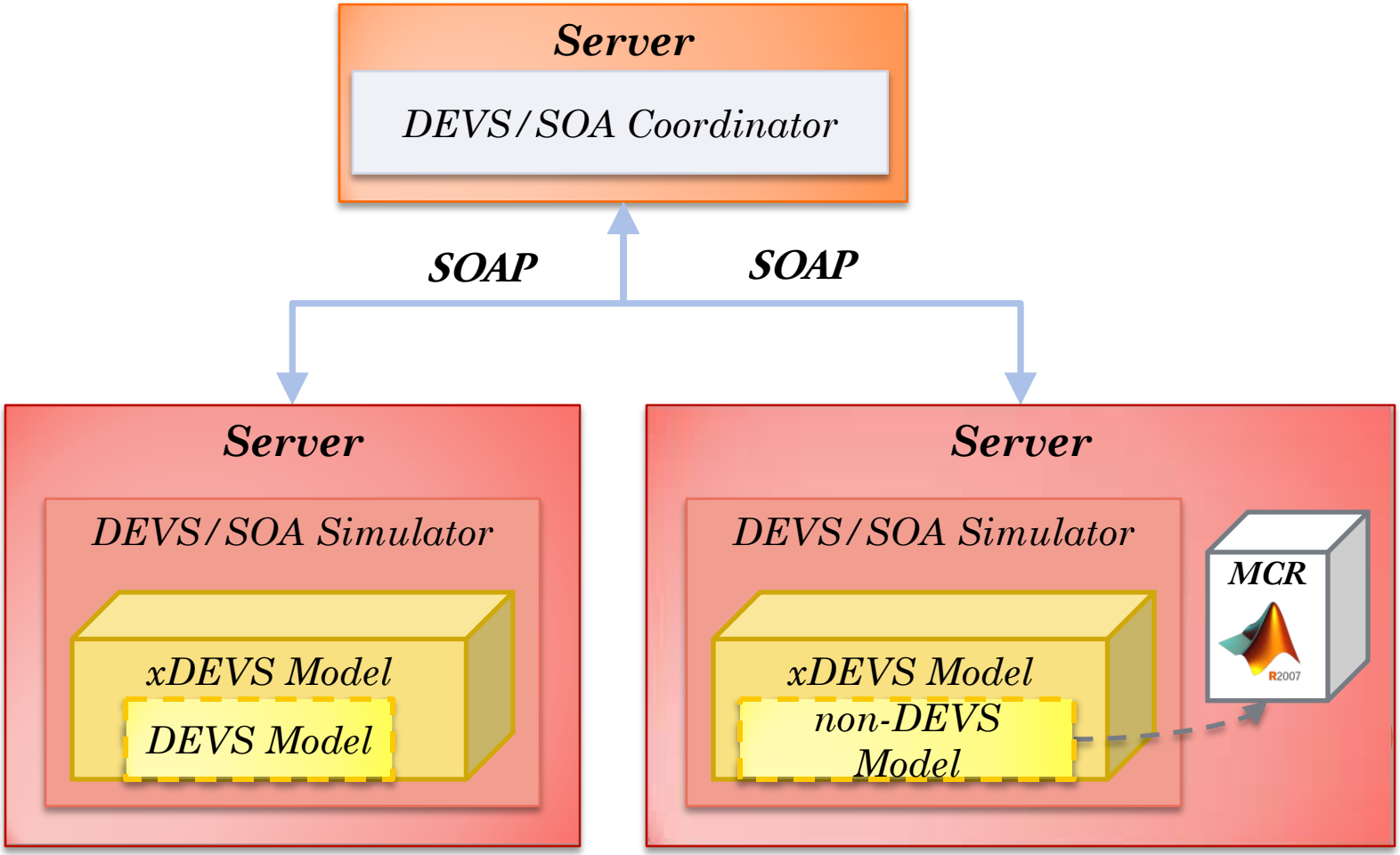
Interoperability between DEVS and non-DEVS models using DEVS/SOA

- *Modeling Approach*



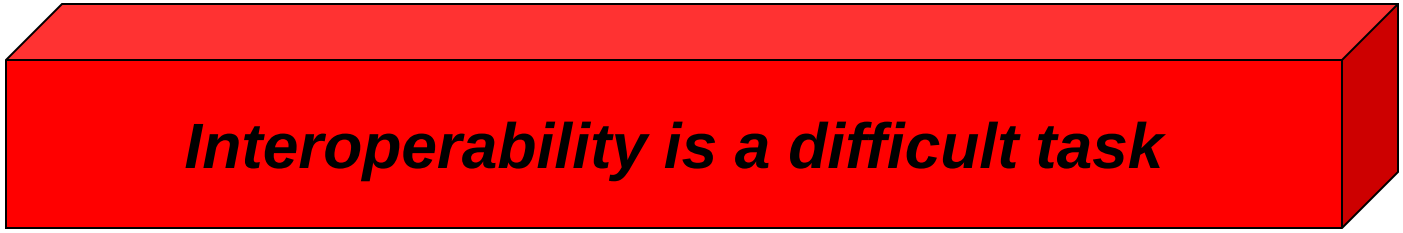
Interoperability between DEVS and non-DEVS models using DEVS/SOA

- *Simulation Approach*



DEVS/SOA - Problems

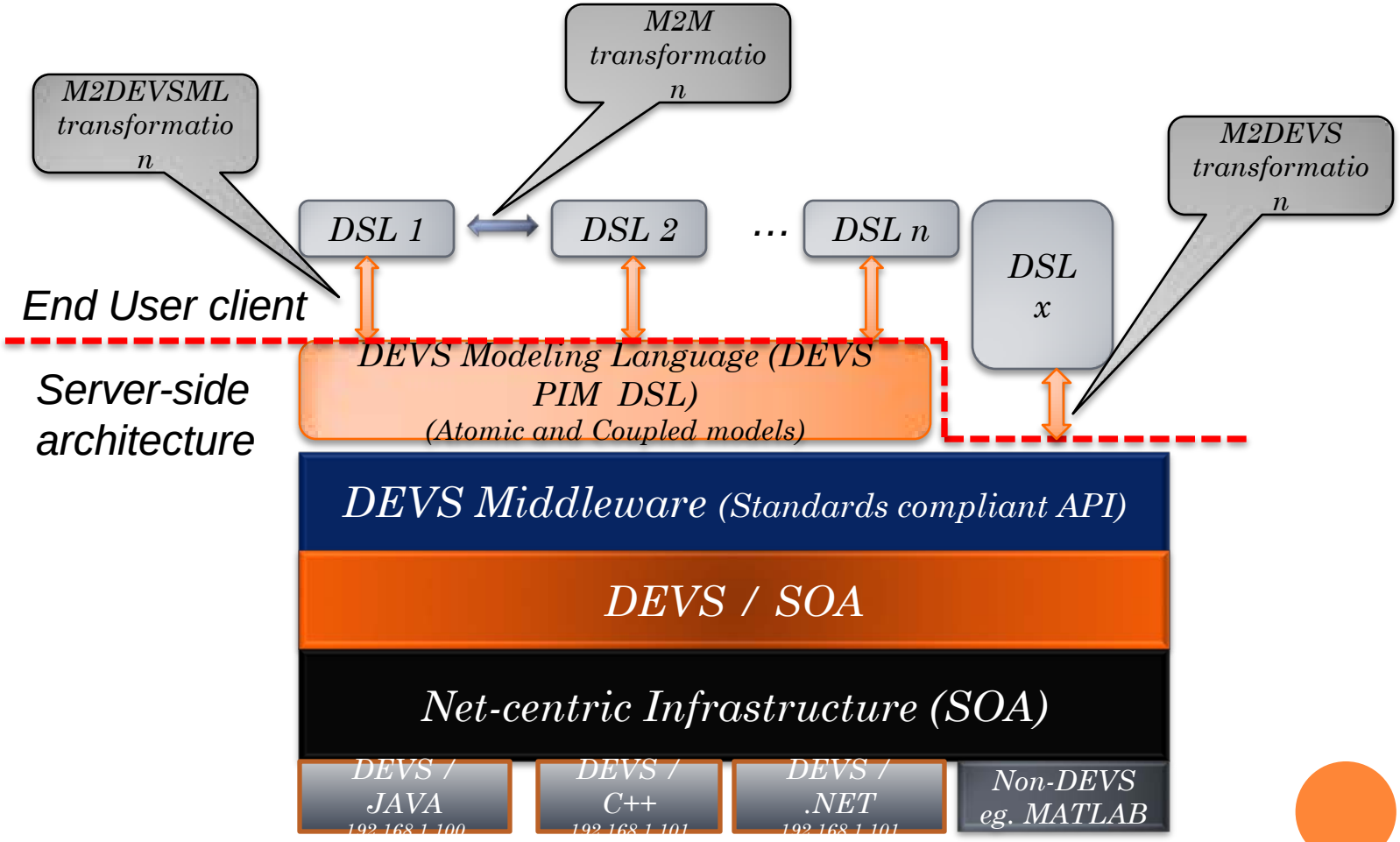
- The state of Web Services is memorized by means of “static containers”
  - It does not work in .NET, for example
- All the servers receive the whole model (java files), where it is compiled
- Messages (events) are serialized in bytes



DEVS/SOA – Solutions
Upload and compile & Message serialization

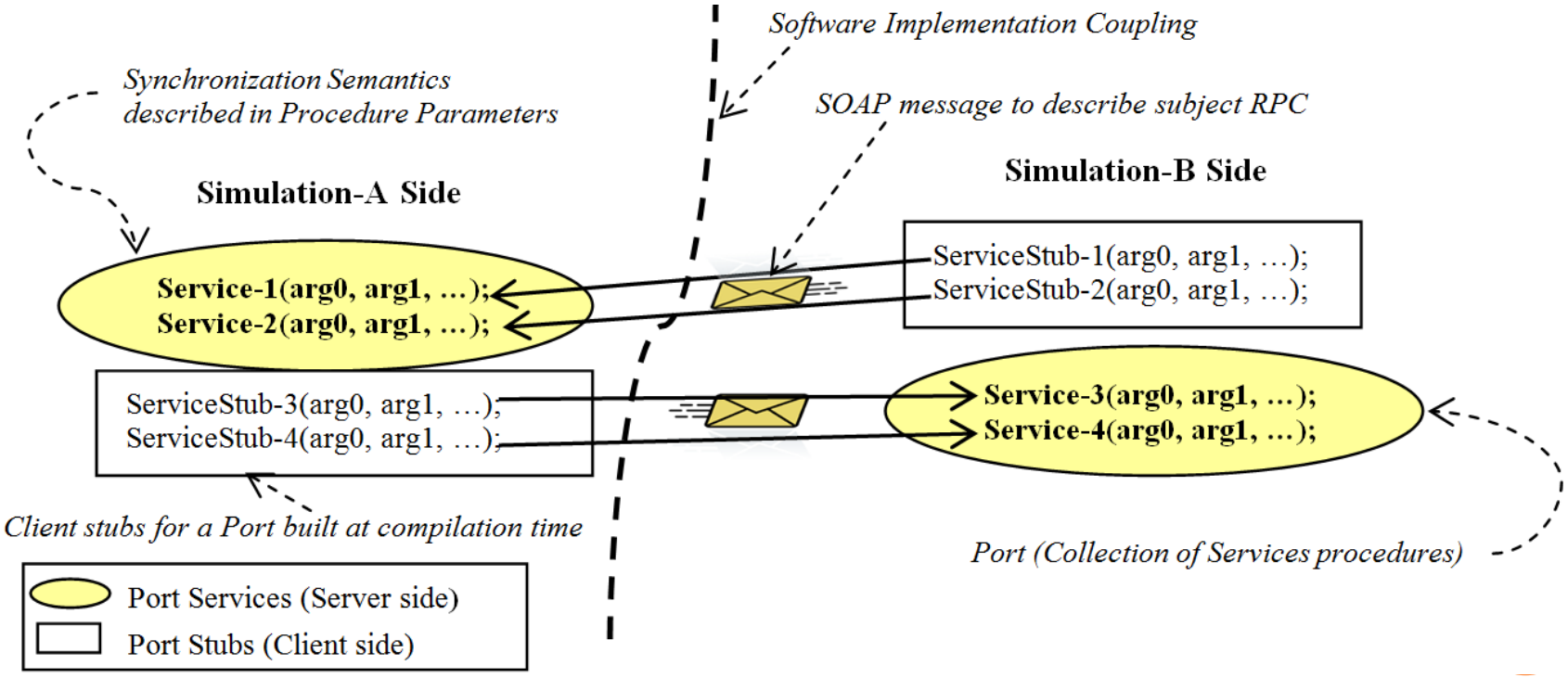
- In addition to upload and compile models, we propose:
  - Simulators as web services (done)
  - Models as web services as well
    - Repositories of models
- Message serialization: WSDL
  - Structure of Messages are defined in WSDL ➔ XML serialization can be done automatically

Model & Simulator Interoperability



Existing Approaches (SOAP WS)

- Simulation Components
  - Communicate with RPC-style
  - RPC is converted to SOAP by WS underlying layer



Existing Approaches (SOAP WS)

- Heterogeneous interface
  - RPCs invented by programmers
- Making interoperability sensitive to changes
- RPCs: difficult to support multiple interoperability protocols.
  - A Port for each group of RPCs
- Difficult to develop Synchronization semantic standards.
  - Standardizing RPCs names and programming parameters
- Services composition does not scale well

RISE Methodologies

- Uses RESTful WS Structural Rules
  - Interoperate in the Web style
- Widely accepted standards: XML, URI, and HTTP.
- Hides implementations (heterogeneity) in resources (resource-oriented):
  - Resources addressed via URI templates.
    - Created by a unique URI instance
  - Resources: classes of services
    - Allow multiple services Support
  - Experiment Blueprint Framework
    - Experiment is made of a number of URIs

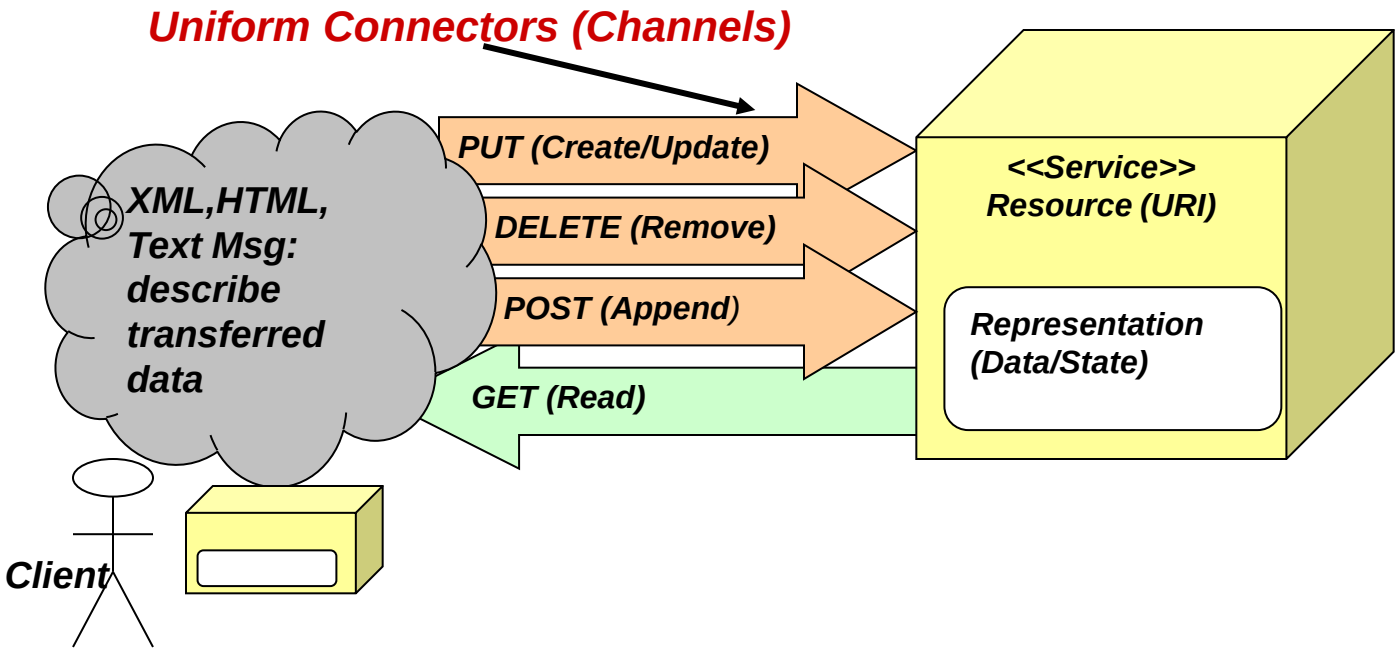
RISE Methodologies (Cont.)

- Exchanged Information structure (syntactic):
  - Simulation semantics expressed in XML messages
  - Multiple-semantic support
    - A system can be implemented to support multiple XML messages
    - allow systems to evolve independently
  - Remote simulation messages aggregation
    - Improve distributed simulation performance

Representational State Transfer (REST)

- Web Style & Principles (reusing existing assets through the Web)
  - Message-oriented
  - Services: self-contained blocks

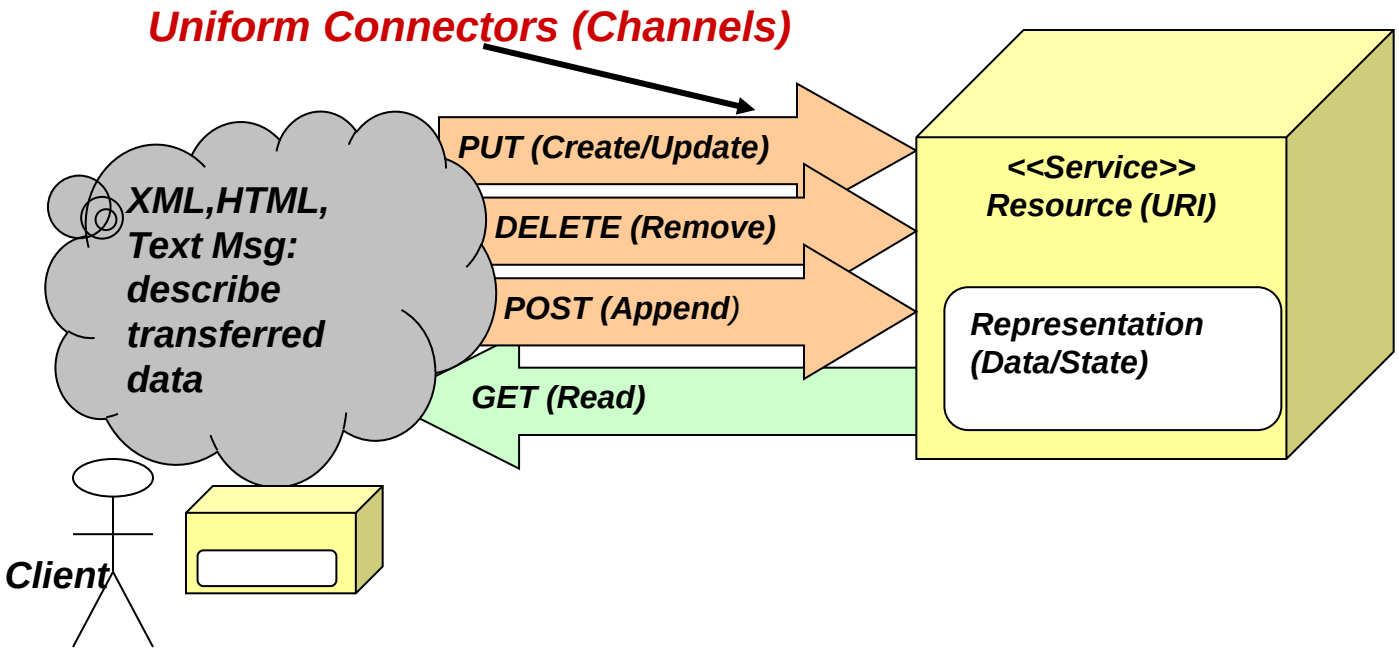
  - Uniform Connectors
  - Resources: uniquely addressed (URI)



Representational State Transfer (REST)

- Web Style & Principles (reusing existing assets through the Web)
  - Message-oriented
  - Services: self-contained blocks

  - Uniform Connectors
  - Resources: uniquely addressed (URI)



Resources

- **Group URL:** http://www.sce.carleton.ca/faculty/wainer/DEVSTD

- **Mailing list**:

SISO Reflector

http://www.sisostds.org/

- **Old posted messages**:

http://groups.yahoo.com/group/DEVSTD

Other resources:

http://www.acims.arizona.edu/

http://www.sce.carleton.ca/faculty/wainer/celldevs/

- **Coordination**: Gabriel A. Wainer. gwainer@sce.carleton.ca