Ptolemy Miniconference Poster Presentations

One Minute Poster Overviews,
Not regular 20 minute presentations

Instructions

Place your poster in **reverse** alphabetical order by last name, so Weber, Matt will be first and we will end with Bagheri, Maryam.

Please just upload one slide.

Pretty Please, just stick to one minute.

Localize the World: A Location Accessor Architecture for the Swarm

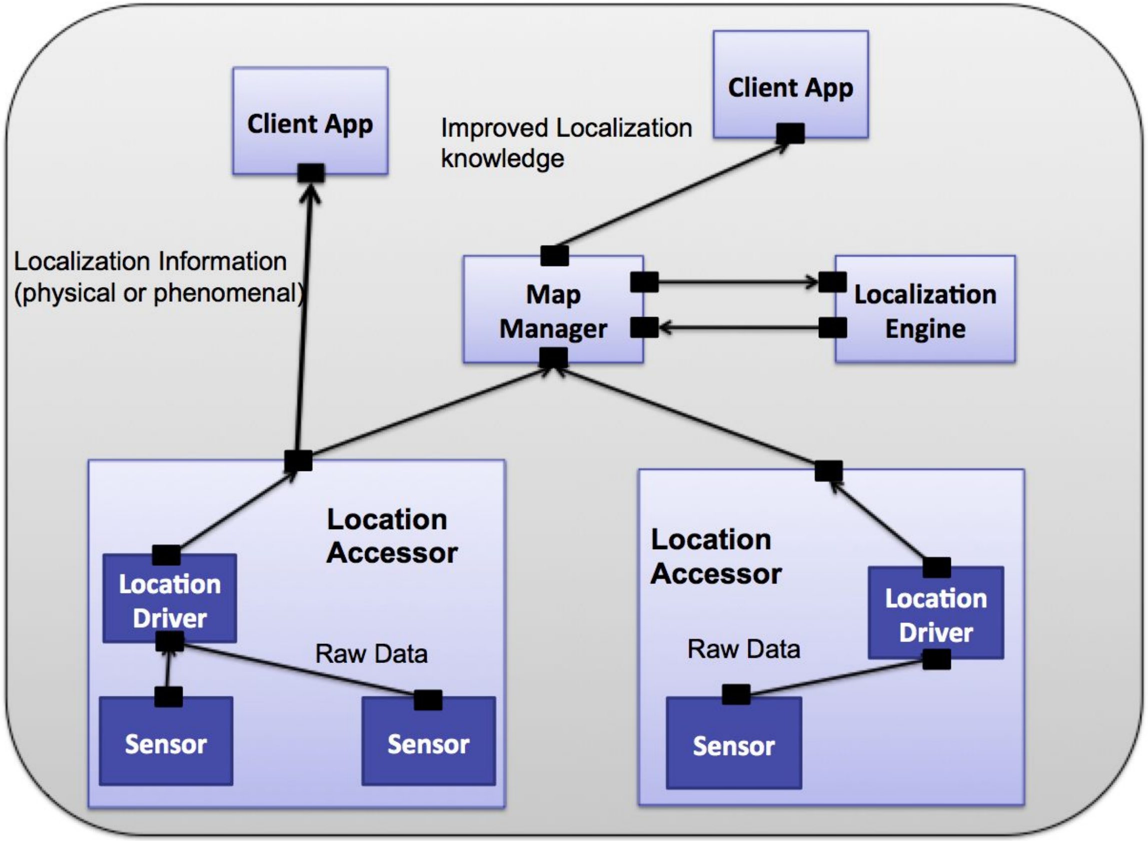
Matt Weber, Edward Lee, UC Berkeley

Goal: Present a common abstract representation of position to swarmlets.

Map Manager
- Maintain relationships between maps
- Evaluate first order logic sentences about position

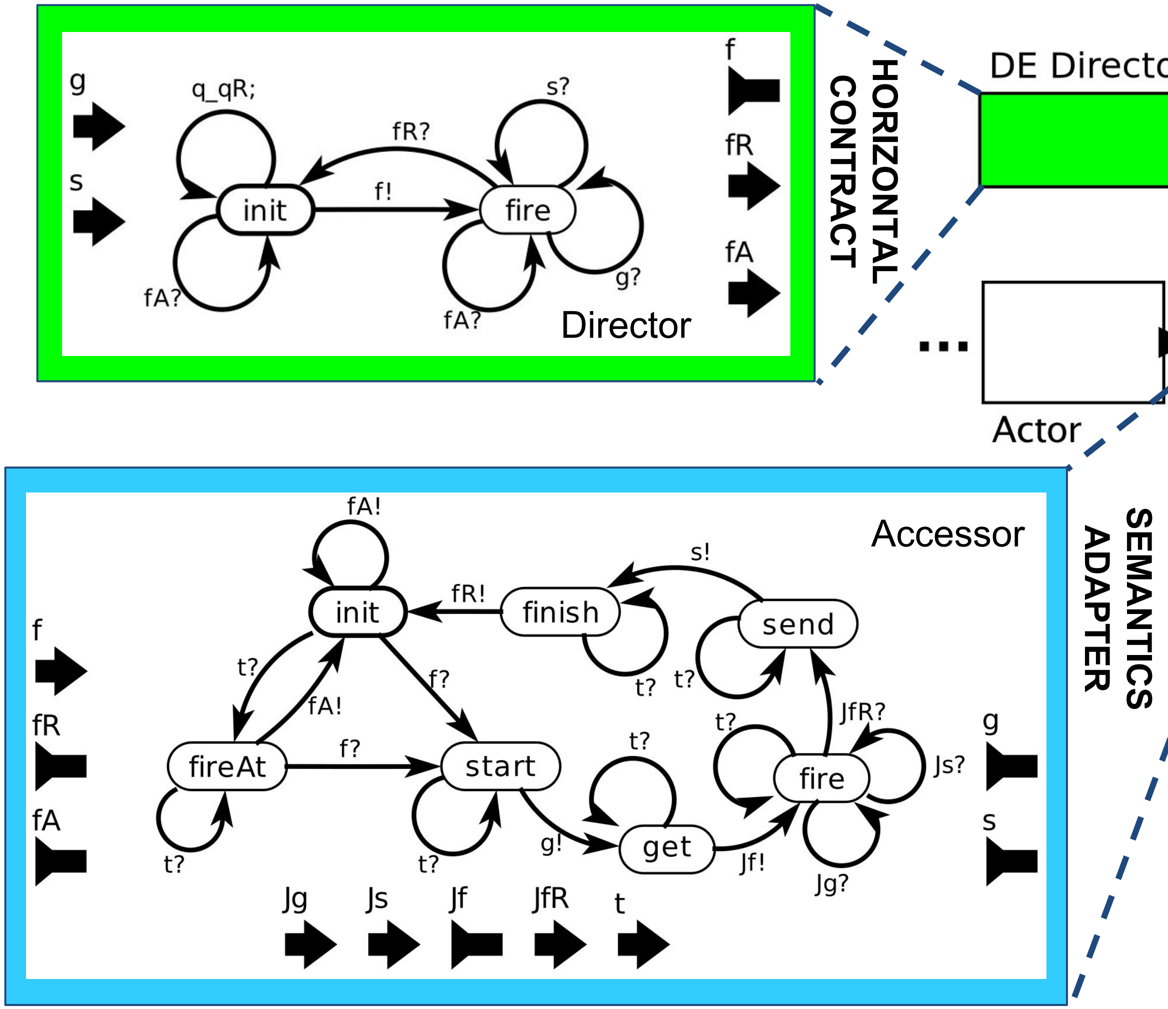
Location Engine
- Automatic sensor fusion
- Check map consistency

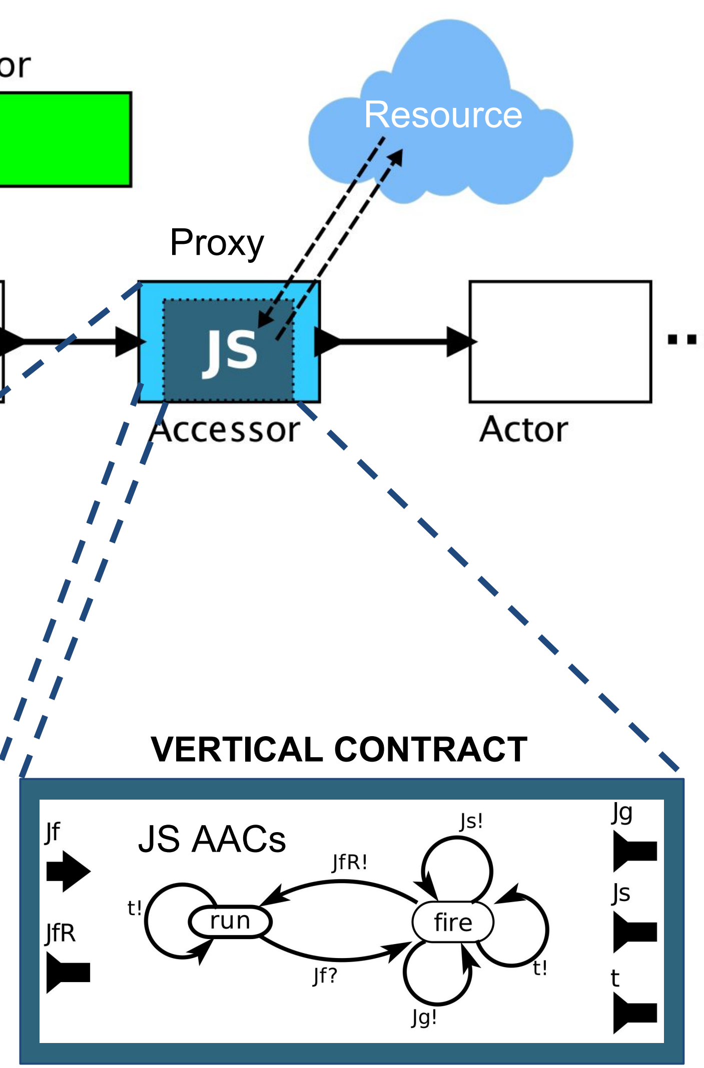
Applications
- Better localization
- Maintain privacy
- Catch liars
- Bound errors



An Interface Theory for the IoT

Marten Lohstroh and Edward A. Lee
University of California, Berkeley





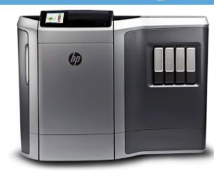
Lohstroh & Lee, “An Interface Theory for the Internet of Things,” Proc. of Software Engineering and Formal Methods (SEFM), York, England, Sept, 2015.

Fast Simulation Techniques for HP Multi Jet FusionTM 3D Printing Technology using Ptolemy II

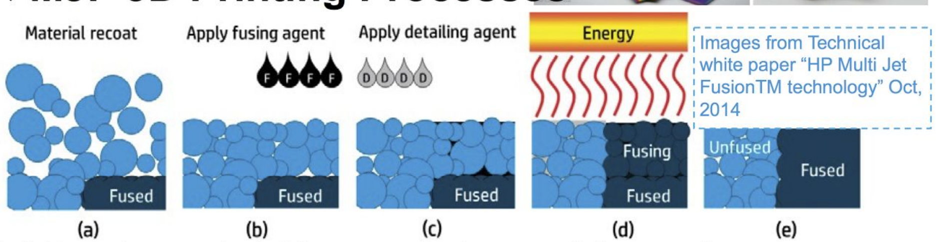
Hokeun Kim and Yan Zhao (HP Labs)

HP Multi Jet Fusion 3D Printing Technology

- Fast and inexpensive technology
- Can provide new levels of quality (different colors, strengths, flexibility, conductivity, etc.)
- Layer by layer production, selectively fusing each powder layer



❖MJF 3D Printing Processes



(a) Powder material is recoated around the work area

(b) Fusing agent is selectively applied to the printing area

(c) Detailing agent is applied where fusing needs to be reduced

(d) The work area is exposed to radiation energy for fusing

(e) Fused area and unfused area after fusing

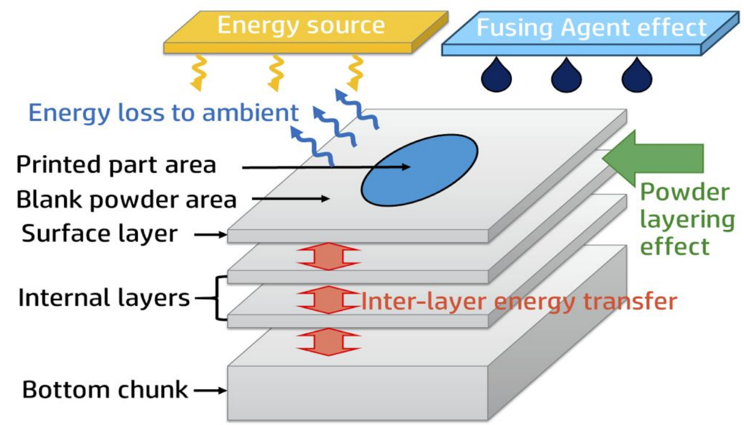
Objectives

❖Process-Level Simulation of 3D Printer

- Predicting printed parts' quality from simulated results
- Providing guidance for development of future materials/processes by exploring different parameters

Modeling Techniques

❖Layer and Area Approximation



❖Use of Empirical Functions

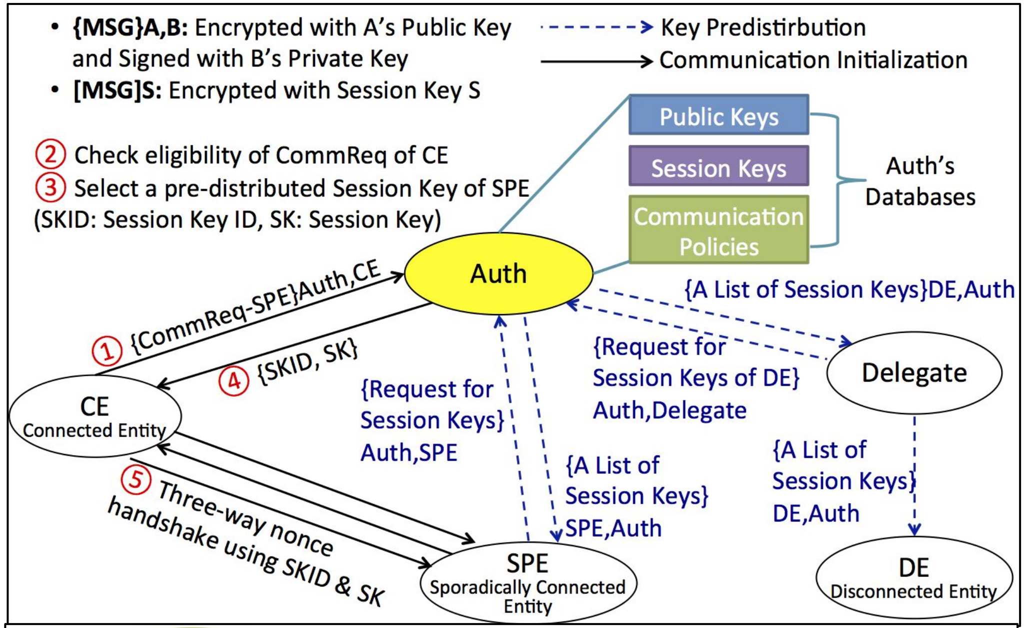
A Secure Network Architecture for the Internet of Things Based on Local Authorization Entities

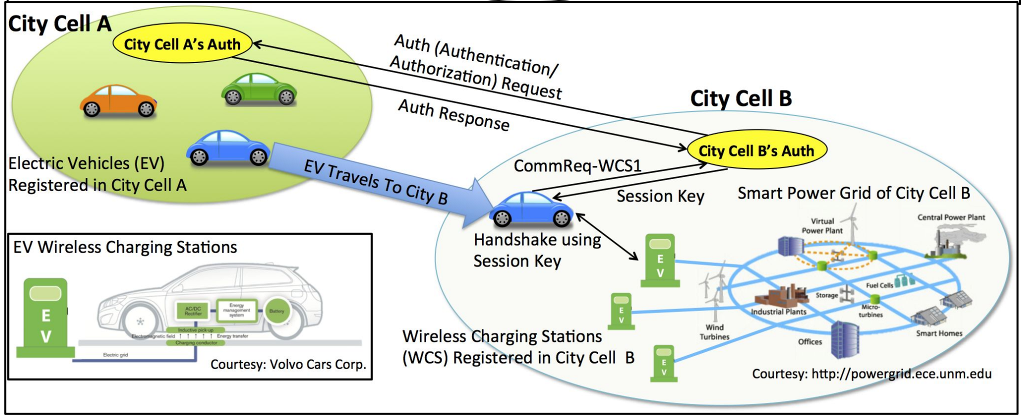
Hokeun Kim, Armin Wasicek and Edward A. Lee

Goal: Design a secure network architecture for the Internet of Things while addressing the IoT-specific challenges

Approach:
1. Use local **auth**orization/ **auth**entication entity, **Auth**
2. Categorize network entities according to their characteristics
3. Use session keys with timeouts that can be predistributed
4. Delegates for devices with limited communication capability

Application Example: Electromobility example involving electric vehicles, wireless charging stations, and smart grid





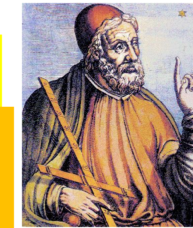
October 16, 2015

Berkeley, CA

Ptolemy Miniconference

FIDE – An FMI Integrated Design Environment

Fabio Cremona, UCB
Marten Lohstroh, UCB
Stavros Tripakis, UCB
Christopher Brooks, UCB
Edward A. Lee, UCB

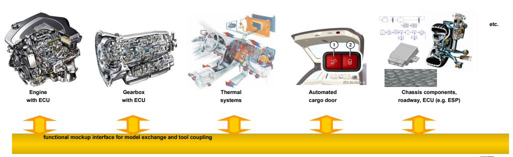


Abstract

This paper presents FIDE, an Integrated Development Environment (IDE) for building applications using Functional Mock-up Units (FMUs) that implement the standardized Functional Mock-up Interface (FMI). FIDE is based on the actor-oriented Ptolemy II framework and leverages its graphical user interface, simulation engine, and code generation feature to let a user arrange a collection of FMUs and compile them into a portable and embeddable executable that efficiently co-simulates the ensemble. The FMUs are orchestrated by a well-vetted implementation of a master algorithm (MA) that deterministically combines discrete and continuous-time dynamics. The ability to handle these interactions correctly hinges on the implementation of extensions to the FMI 2.0 standard. We explain the extensions, outline the architecture of FIDE, and show its use on a particularly challenging example that cannot be handled without the proposed extensions to FMI 2.0 for co-simulation.

Functional Mockup Interface (FMI)

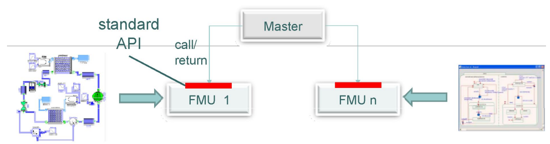
FMI is a promising standard for model-exchange and co-simulation.



In FMI-ME, an FMU only declares a set of variables, equations, and optional data such as parameter tables or user interface features. This mode provides a common format for the exchange of components across different simulation tools. In FMI-CS, on the other hand, an FMU is a self-contained object that besides the model description also includes the simulation engine provided by the design environment in which it was created. For example, a co-simulation FMU may provide the functionality of an executable Simulink model.

Master-Slave architecture:

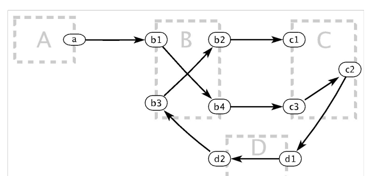
- A master coordinate the simulation
- Many slaves (FMUs): **"black boxes"** implementing a certain API. These correspond to sub-models exported by various modeling tools



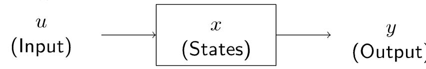
FMI: limitations and extension

- Super-Dense time?
- Data-type to represent time?
- Absent values?
- FMU contracts for efficient rollback?

FMUs evaluation order



- `get()` known outputs.
- `set()` dependent inputs → repeat (while respecting the dependencies), until all I/O ports are set.
- `doStep()` to update the state of all FMUs

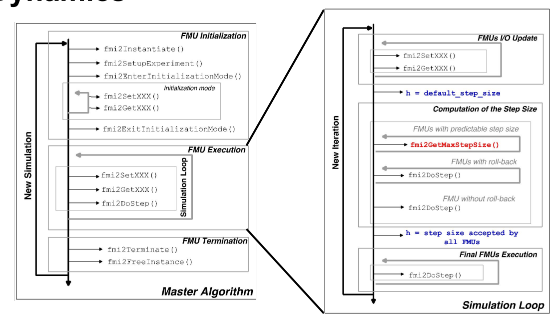


$\texttt{set}_c(s, u, v) \mapsto s$

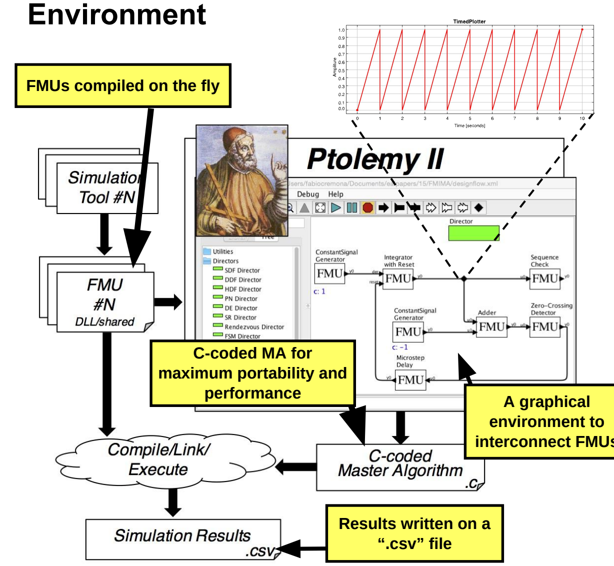
$\texttt{get}_c(s, y) \mapsto v$

$\texttt{doStep}_c(s, h) \mapsto (s', h')$

A Master Algorithm for DE and CT Dynamics



FIDE – An FMI Integrated Design Environment



Organization *Sponsorship*

The TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA; the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies); the National Science Foundation (NSF) awards #1446619 (Mathematical Theory of CPS), #1329759 (COSMOI), #0931843 (ActionWebs), #0720882 (CSR-EHS: PRET), #1035672 (CPS: Medium: Timing Centric Software); the Naval Research Laboratory (NRL #N0013-12-1-G015); and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, supported by the following companies: Denso, IHI, National, Instruments and Toyota.

Industrial Cyber-Physical Systems

College Of Engineering, UC Berkeley

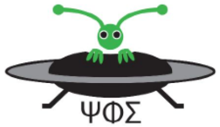
CyPhySim

A Cyber-Physical Systems Simulator

Christopher Brooks, UCB
Fabio Cremona, UCB
Edward A. Lee, UCB
David Lorenzetti, LBNL
Thierry S. Nouidui, LBNL
Michael Wetter, LBNL

http://cyphysim.org

CyPhySim



CyPhySim is an open-source simulator for cyber-physical systems. The simulator provides a graphical editor, an XML file syntax for models, and an open API for programmatic construction of models.

CyPhySim supports the following Models of Computation:

- Discrete Events simulation
- Quantized-State Systems (QSS) simulation
- Continuous time (Runge-Kutta) simulation
- Discrete time simulation
- Modal Models
- Functional Mockup Interface (FMI)
- Algebraic loop solvers

Discrete-Event Simulation

In the style of discrete-event (DE) modeling realized in CyPhySim, a model is a network of actors with input and output ports. The actors send each other time-stamped events, and the simulation processes these events in time stamp order.

This style of DE is widely used for simulation of large, complex systems. CyPhySim builds on the particular implementation in Ptolemy II, which has a sound, deterministic semantics.

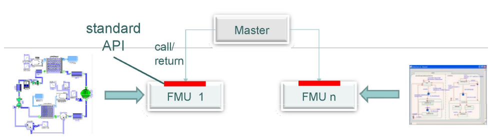
Model of Time

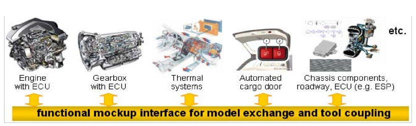
For models that mix discrete and continuous behaviors, it is well established that the model of time that is used must support sequences of causally-related instantaneous events **CyPhySim** uses superdense time.

Functional Mockup Interface (FMI)

FMI is a promising standard for model-exchange and co-simulation.

•Model exchange across different engineering departments and across vendors
•Tool coupling for co-simulation





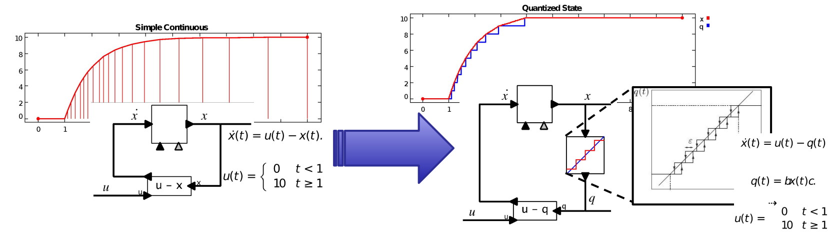
Picture from "Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models"

Master-Slave architecture:

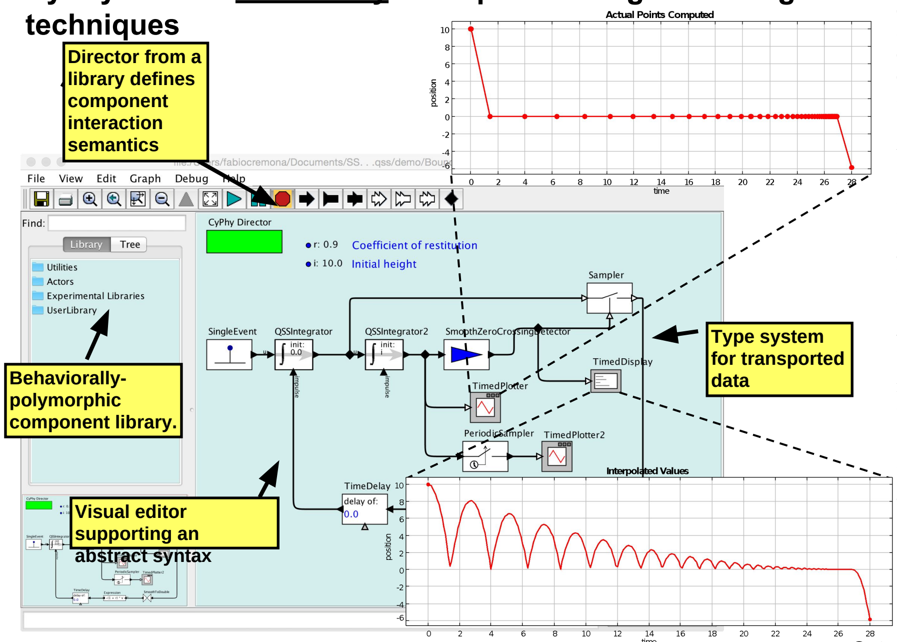
•A master coordinate the simulation
•Many slaves (FMUs): **"black boxes"** implementing a certain API. These correspond to sub-models exported by various modeling tools

Quantized-state Systems

A relatively recent development in numerical simulation of ordinary differential equations is the emergence of so-called *quantized-state systems* (QSS). In a classical ODE simulator, a step-size control algorithm determines sample times, and a sample value is computed at those times for all states in the model. In a QSS simulator, each state has its own sample times, and samples are processed using a DE simulation engine in time-stamp order. The sample time of each state is determined by quantizing the value of each state and producing samples only when the value changes by a pre-determined tolerance, called the quantum.



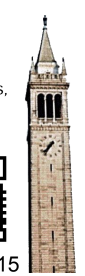
CyPhySim is a Laboratory for experimenting with design techniques



Organization

Sponsorship

The TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA; the iCyPhy Research Center (Industrial Cyber-Physical Systems, supported by IBM and United Technologies); the National Science Foundation (NSF) awards #1446619 (Mathematical Theory of CPS), #1329759 (COSMOI), #0931843 (ActionWebs), #0720882 (CSR-EHS: PRET), #1035672 (CPS: Medium: Timing Centric Software); the Naval Research Laboratory (NRL #N0013-12-1-G015); and the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, supported by the following companies: Denso, IHI, National Instruments, and Toyota.



October 16, 2015