Model-Driven Development of i-DSML Execution Engines

Gustavo Sousa and Fábio M. Costa (UFG-Brazil)
Peter J. Clarke (FIU), Andrew A. Allen (GSU)

Models@RunTime 2012

Introduction

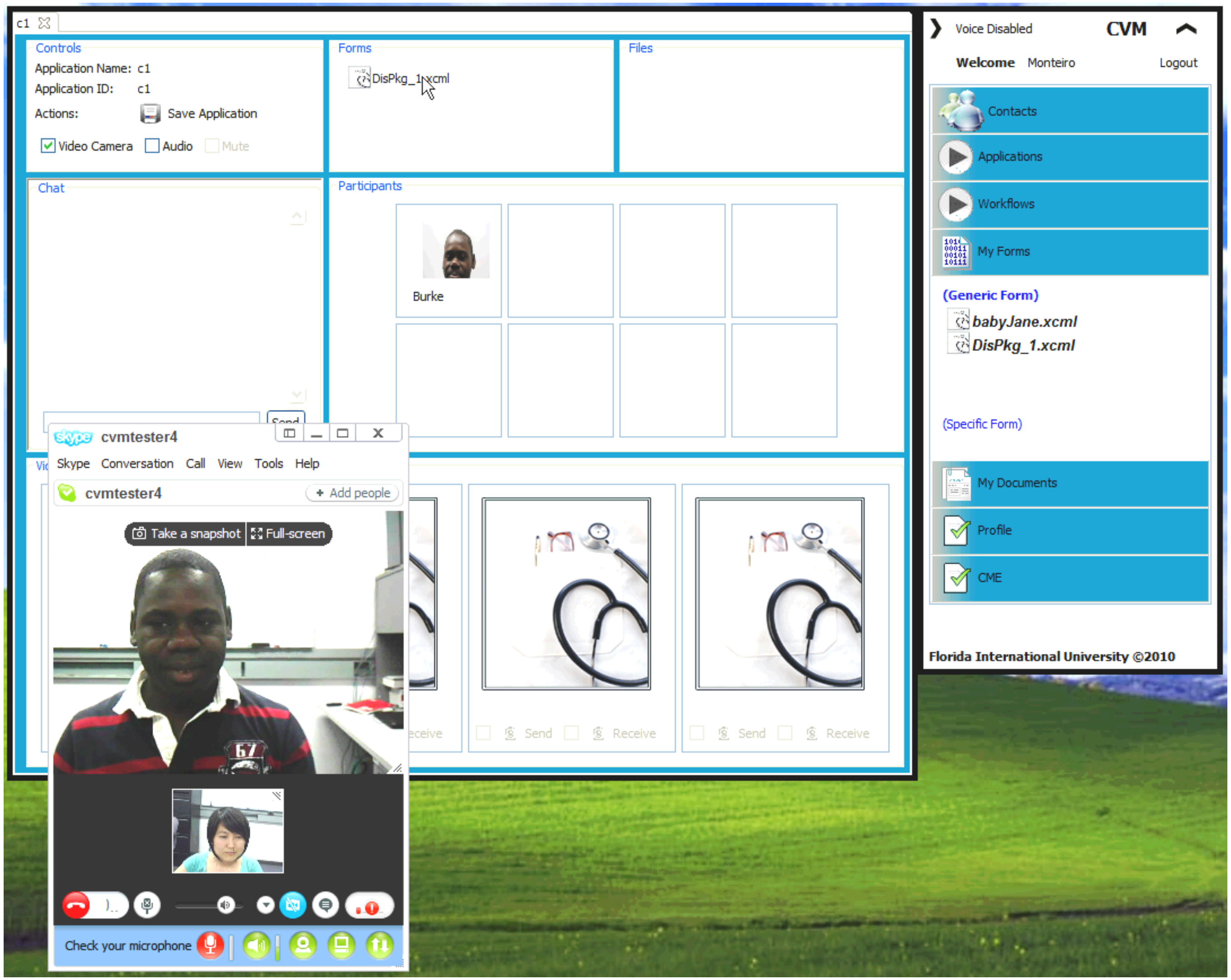
- *A DSML is a special-purpose graphic specification language that is tailored for a particular domain*.
- Two views of DSMLs
  - automating the generation of source code from DSMs
  - interpreting DSMs at runtime to realize user requirements (Interpreted-DSMLs or i-DSMLs)
- An i-DSML requires an execution engine that can interpret models at runtime

Introduction cont

We are currently developing two i-DSMLs:

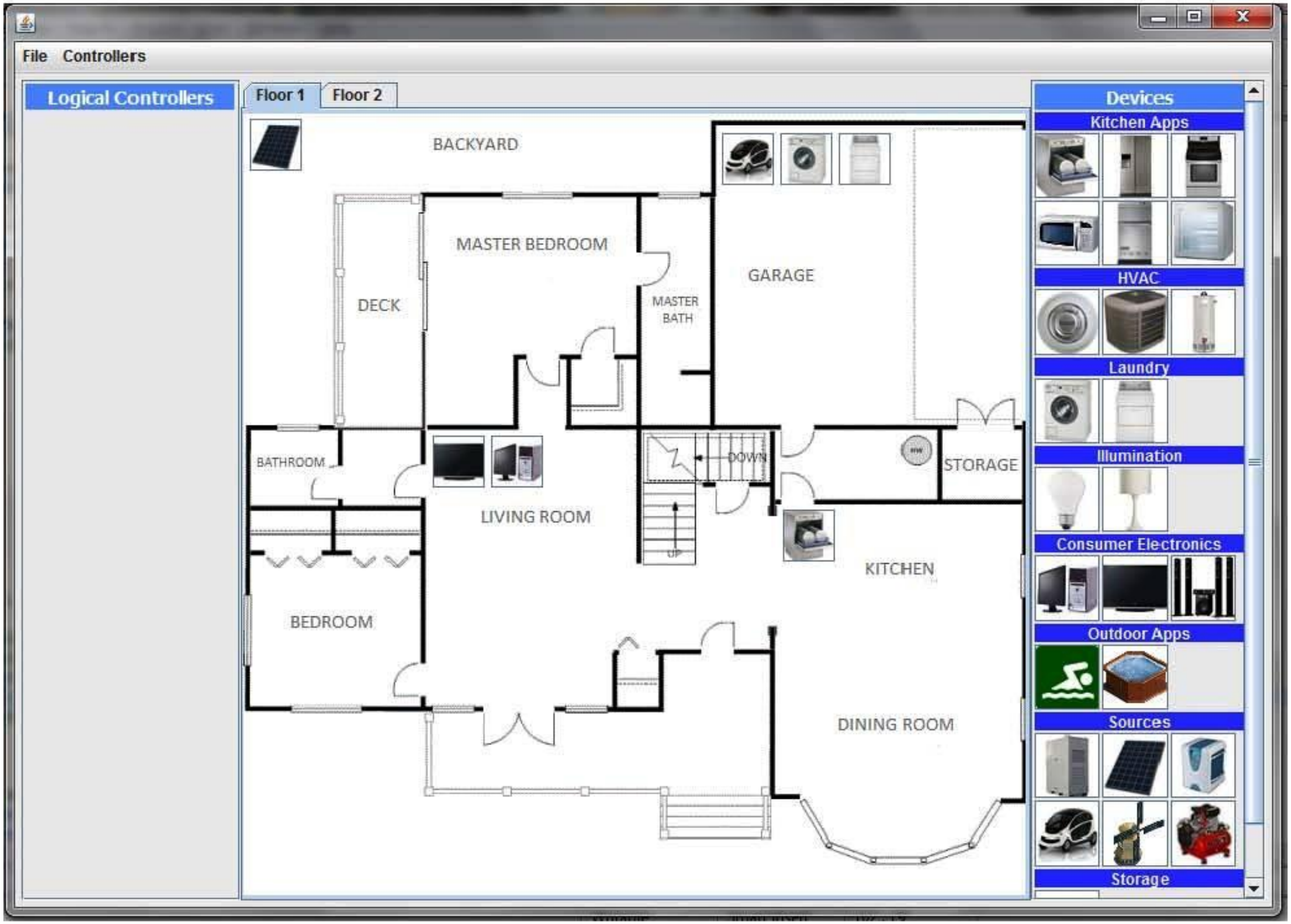
- *Communication Modeling Language (CML)* – used to specify and realize user-defined communication, e.g., send a patient record to doctors in a AV communication
- *Microgrid Modeling Language (MGridML)* – used to specify and realized microgrid energy management scenarios, e.g., apply the Summer energy management model to the current system settings

Introduction cont



User Interface for CML execution engine

Introduction cont



User Interface for MGridML execution engine

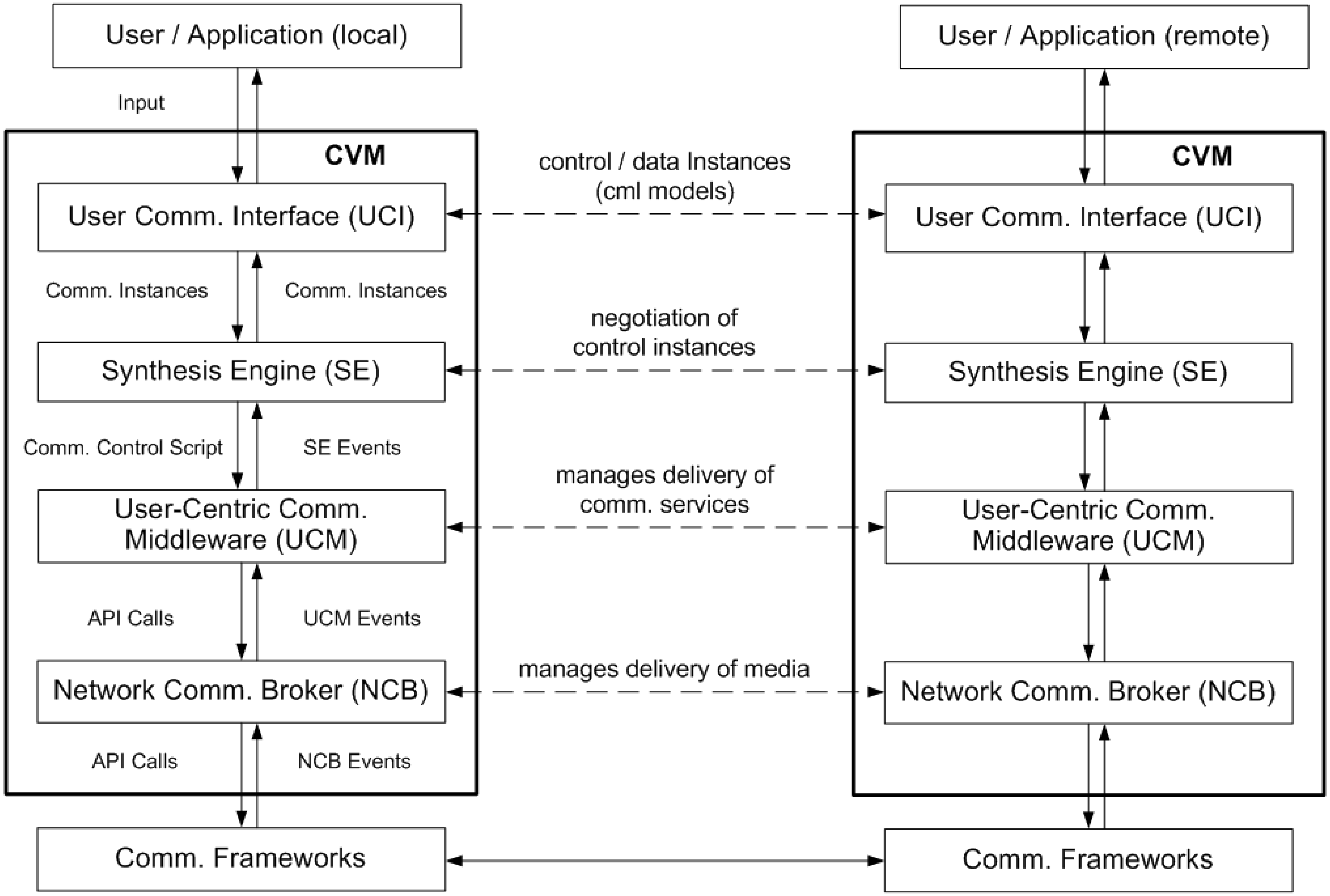
Outline

- Execution Engines
- Problem
- Approach
- Broker Layer MetaModel
- Instance of Broker Layer
- Related Work
- Conclusion

Execution Engines

- Requirements:
  - Interpret user-defined i-DSML models at runtime
  - Use semantics based on changes to i-DSML models at runtime.
  - Apply policies to i-DSML models
- Two execution engines are currently under development:
  - Communication Virtual Machine (CVM)
  - Microgrid Virtual Machine (MGridVM)

CVM Structure



(e.g., Skype, Google Talk, Asterisk)

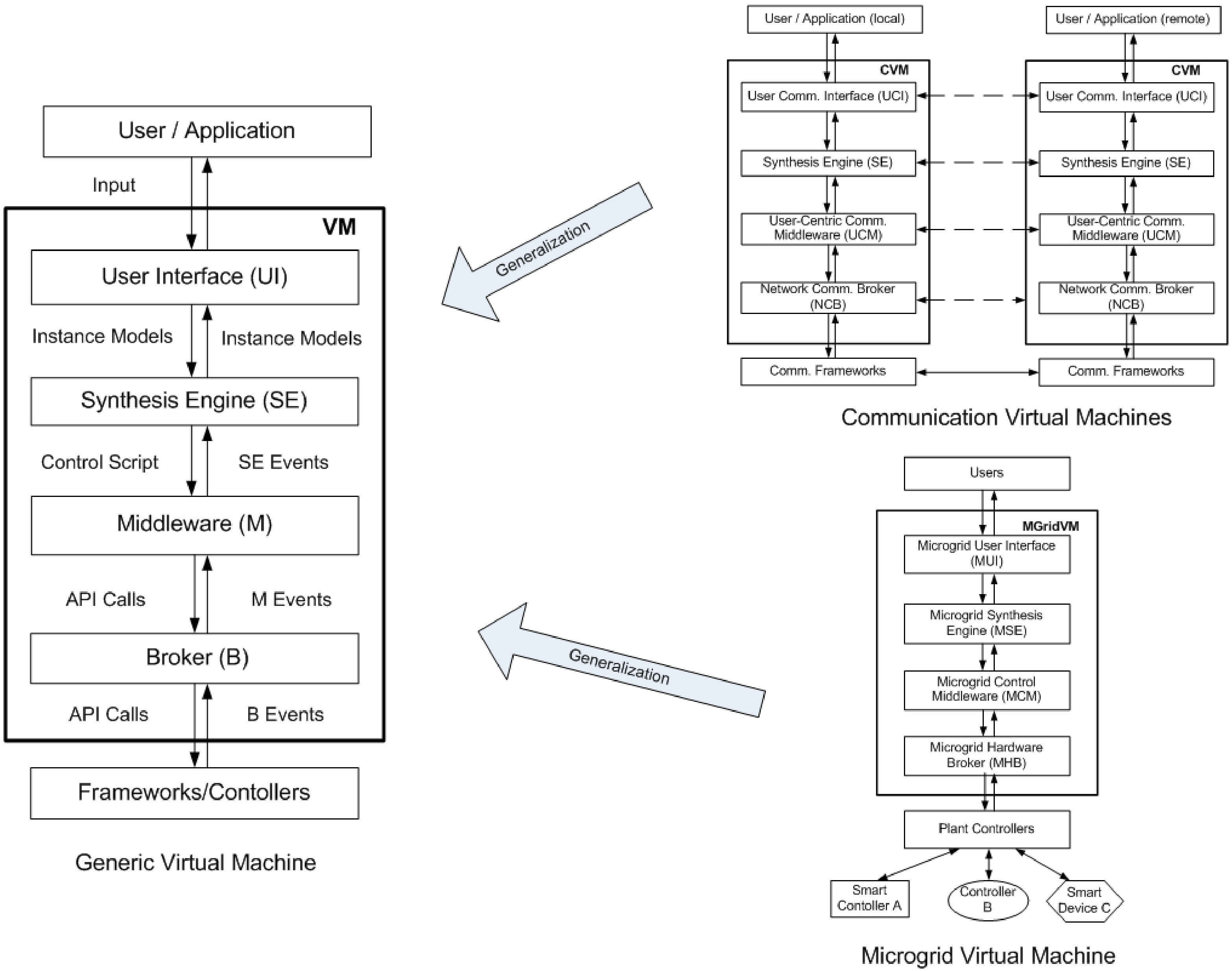
Problem

- Each time a new i-DSML is created for a different domain a substantial re-implementation of the execution engine has to take place.

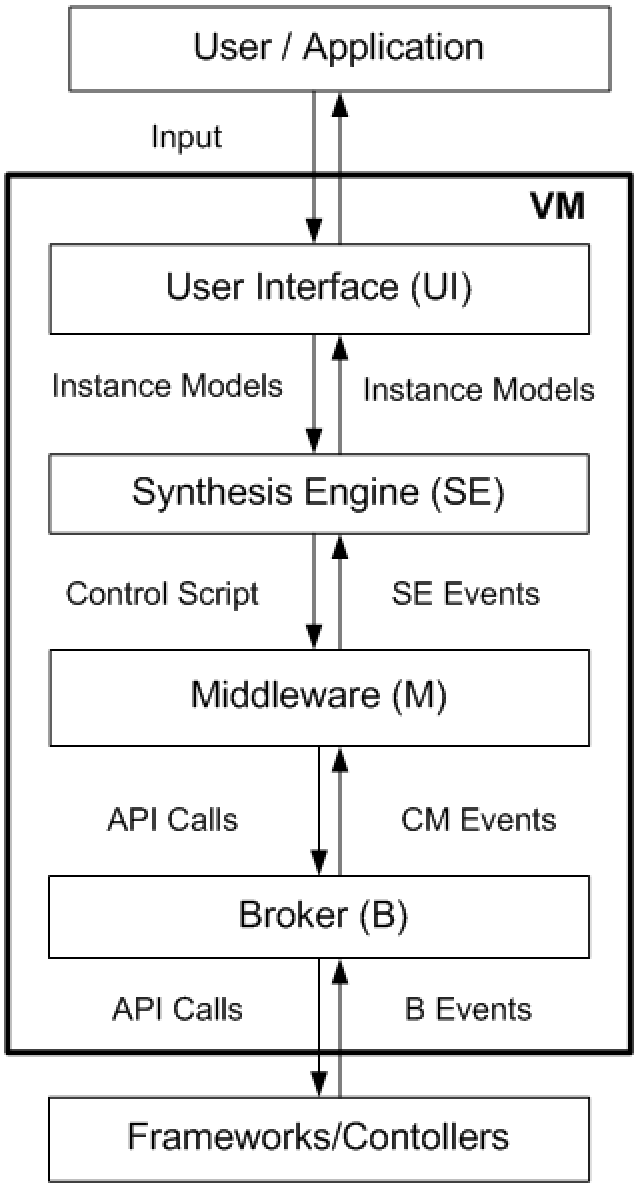
Approach

1. Identify a generic architecture for the execution engine for a class of i-DSMLs
2. Specialization is achieved by modeling in conformance to a given metamodel
3. Metamodel encompasses constructs related to the operations needed for the execution of a class of i-DSMLs

1. Generic Architecture



Execution Engine Structure



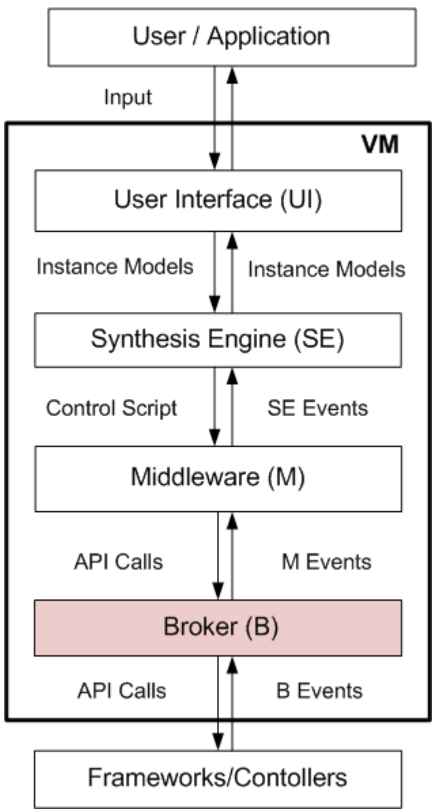
UI - supports creation of DSMs

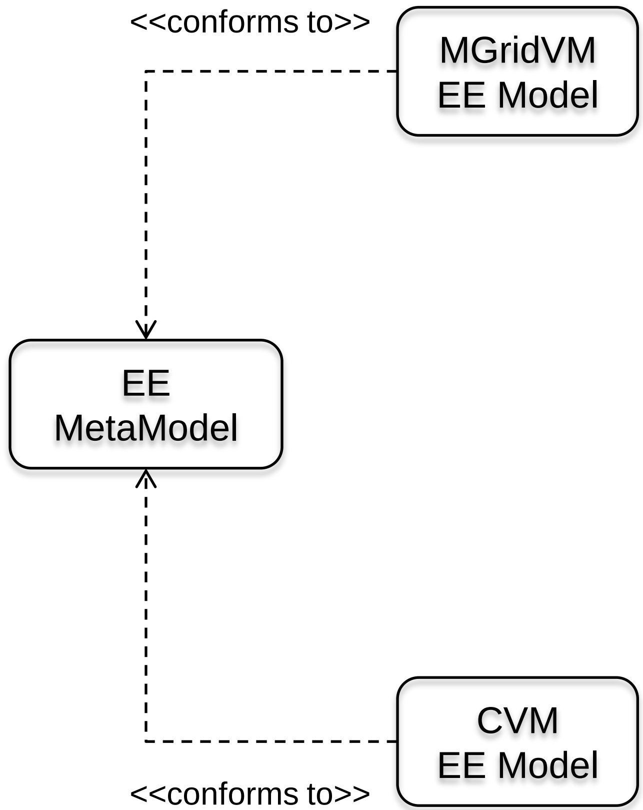
SE – synthesizes model instances generating control scripts for *M*

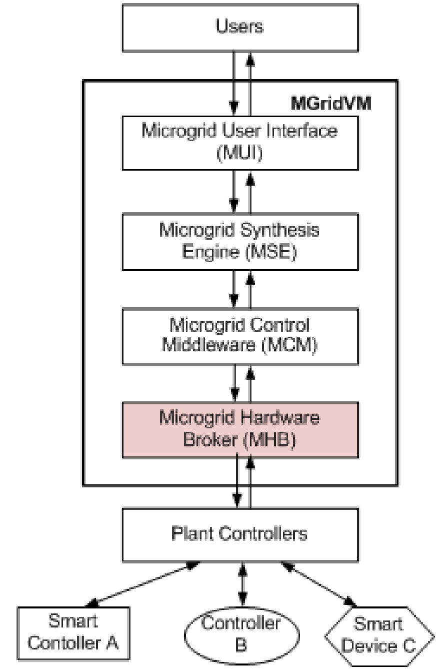
M – executes the control scripts to manage and coordinate the delivery of domain services

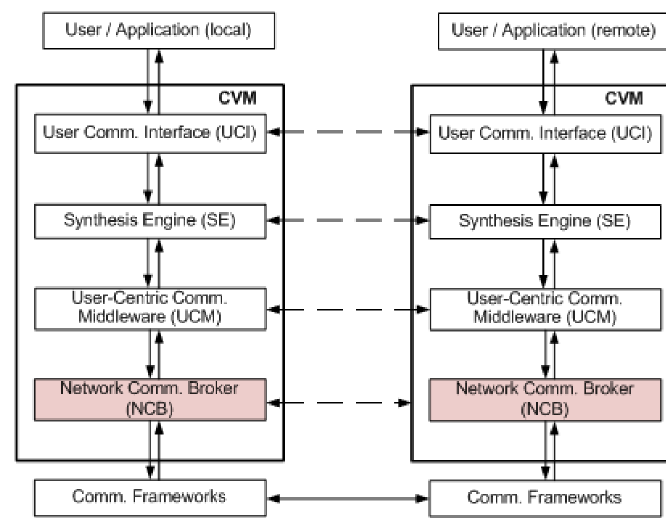
B – provides an independent API to *M* and interfaces with the underlying frameworks and controllers to realize the services

2. Specialization





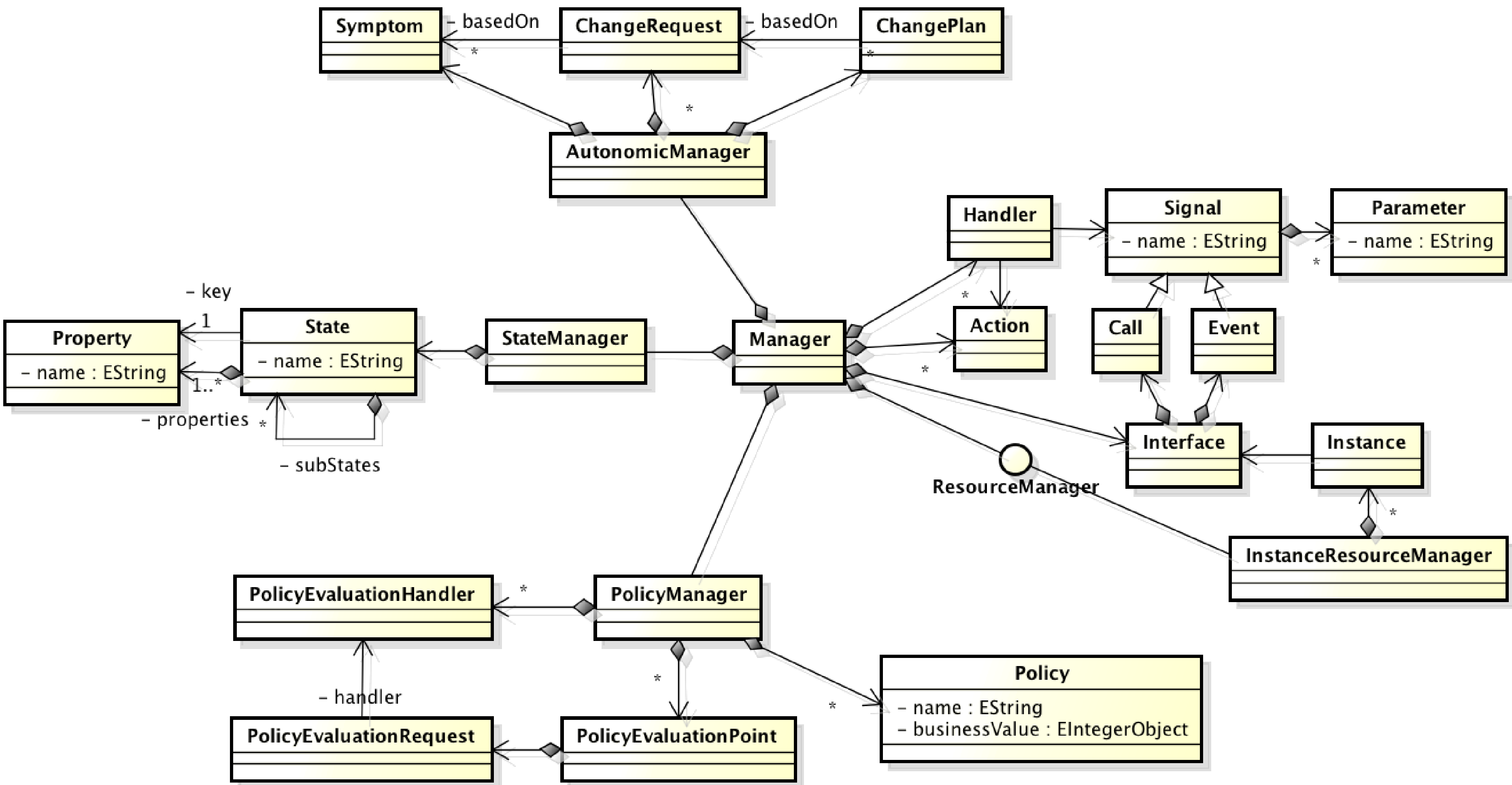




Broker layer

- Provide a uniform interface over resources
- Abstract details in the setup, selection and maintenance of resources
- Automatically identify situations that require *runtime adaptation* and performs them transparently
- Behavior of the *Broker* layer is defined by the way it handles *calls* from upper layer and *events* from resources

Broker layer metamodel



Broker Layer Metamodel cont

- *Interface* describes interfaces of managers and resources
  - Groups a set of calls and events that may be signaled
- *Handlers* define an **action** to be taken:
  - Calls a resource
  - Generates an event to upper layers
  - Executes a sequence of other actions
  - Executes a macro (which may access resources/state)

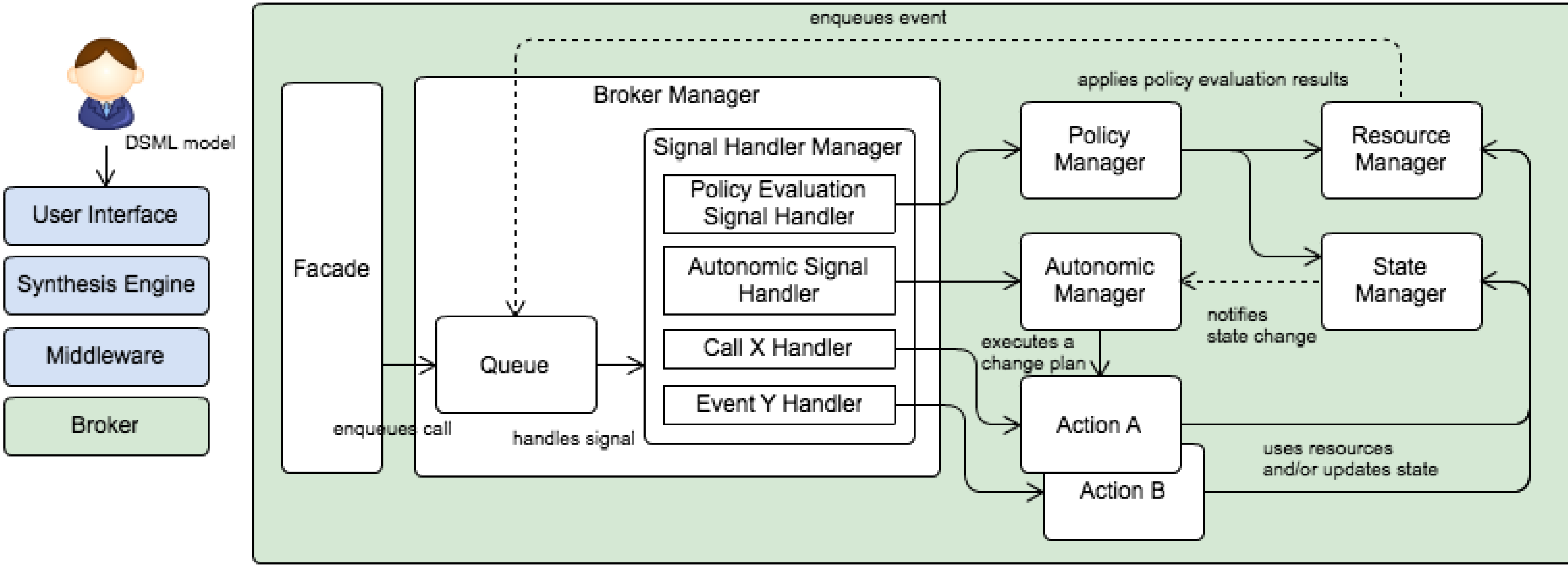
Broker Layer Metamodel cont

- *Resource Management* describes the interface of the managed resources
- *Autonomic Manager* coordinates elements related to autonomic management using MAPE e.g. *Symptom*, *ChangeRequest*, *ChangePlan*
- *PolicyManager* groups policies abstractions related to the definition of policies and their evaluation

Broker Layer Execution Environment

- Metamodel enables the definition of a DSML for the Broker layer
- Provides operational semantics to the Broker layer metamodel
- Loads a Broker layer model and executes accordingly
- Provides a library for integrating resource implementations to the execution environment

Broker Layer Execution Environment cont



CVM - Network Communication Broker

- Provides an uniform API over a set of communication frameworks
- Abstracts communication framework selection, setup, monitoring, replacement and recovery
- Note: the complete model for NCB is too big and it is not feasible to represent it graphically

NCB - Example Scenario

- User requests an audio communication with another party
- A session is initially established using Framework 1
- A failure is simulated in Framework 1
- NCB automatically switches the session to be established using Framework 2

Results for Scenario

- NCB Performance

Original NCB		Modeled NCB	
Avg (ms)	Std Dev	Avg (ms)	Std (ms)
1369.29	51.60	3058.53	149.70

- Substantial performance loss
  - Simple scenarios (significant loading overhead)
  - Evaluation of expressions and parameter bindings

Related Work

- Bryant et al. 2011
  - MDD related to DSMLs focuses on the generation of tools
- Jezequel et al. 2009 (Kermeta)
  - Promotes weaving behavior into the metamodels as a way of defining semantics
  - Semantics are described using an action language
- Chen et al. 2005
  - Use semantic units to define the semantics

Conclusions

- Introduced the notion of i-DSMLs and their execution engines
- Briefly describe an approach for developing the execution engines for a class of i-DSMLs
- Create a metamodel for one layer (Broker) in the execution engine
- Instantiated the metamodel of the Broker to produce the Network Communication Broker

Acknowledgements

- This work was partly supported by the Capes Foundation, Brazil, Proc. 0759-11-2 and FAPEG

Thanks

Questions???