Interoperability 101
ISO/IEC 18012-2 Implementation Example

Ron Ambrosio, IBM T.J. Watson Research Center
Ron Melton, Pacific Northwest National Laboratory

#GridInterop

Approaches to Interoperability

- Adopt a single system as the agreed-upon universal standard
  - Minimizes ability for product differentiation between manufacturers
  - May not adequately address specific regional differences/requirements

OR

- Develop system-to-system translations between all target systems
  - Could easily degenerate into an n2 translation situation
  - Dependence on protocol translation can make the interoperability framework "brittle" – very sensitive to minor changes in underlying systems

OR

- Define a meta-framework to facilitate multi-system solutions without requiring specific system-to-system protocol translations
  - Address unambiguous data mapping/translation
  - Adopt an approach that is independent of the underlying protocols

Guiding Architectural Principles for ISO/IEC 18012

- Two communities of developers being supported:
  - Object/device/service developers ("building the widgets")
  - Solution builders/integrators ("composing the widgets into solutions")
- Maintain separation of:
  - Application object abstraction from application object implementation
  - Logical application topology from physical device/network topology
- Maximize solution correctness and efficiency via both the development and runtime environments
- Treat time as a fundamental primitive in the programming model
- Enable higher-level abstraction and integration of Operations Domain systems and components through encapsulation
  - Accommodate heterogeneity rather than eliminate it
  - Minimize impact on existing Operations Domain systems and skills

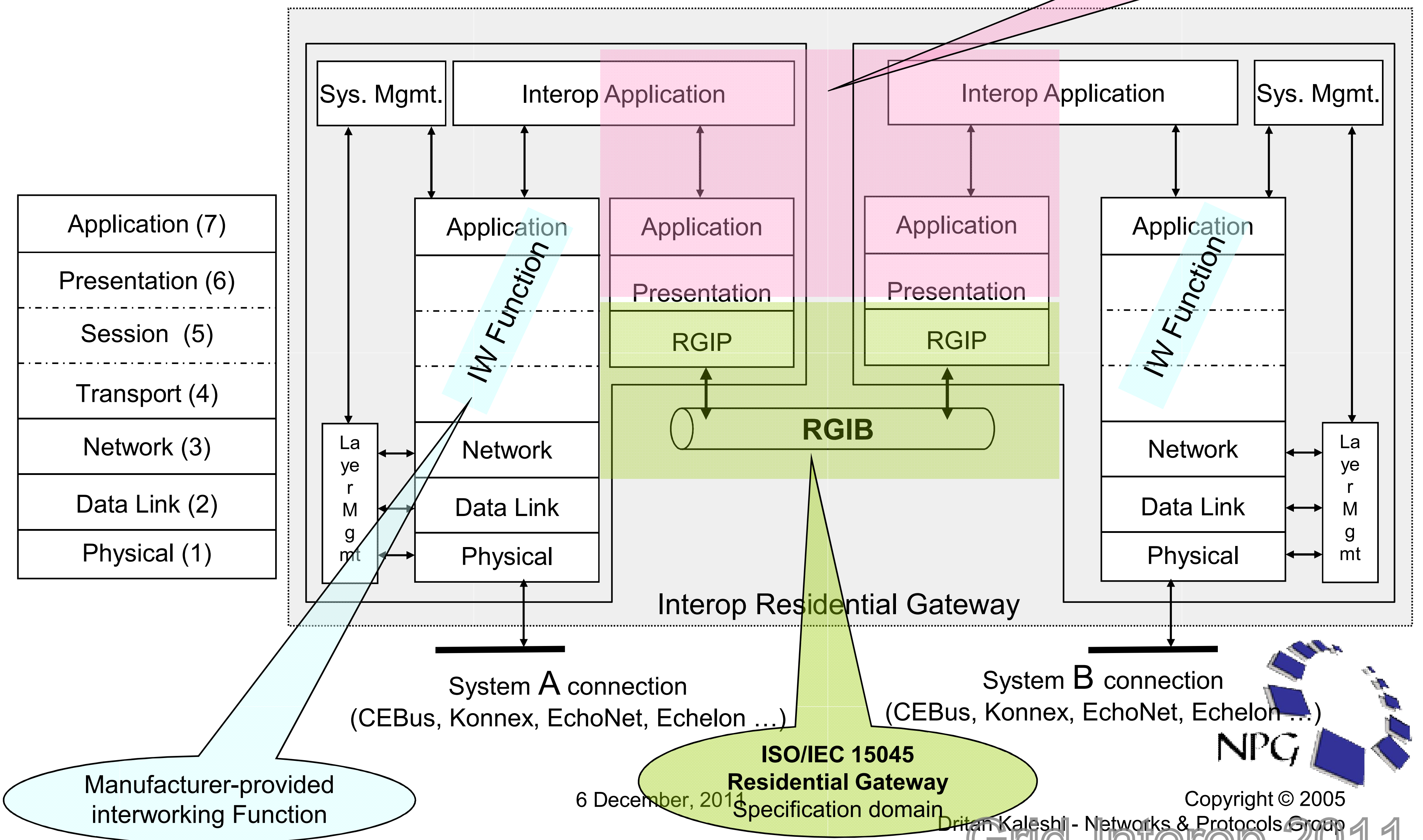
Assumptions & Requirements

- Goal is to support multi-system installations
  - Must address product-level interoperability between specific products/systems and the Interoperability Framework
    - Requirement for the interoperability framework to establish unambiguous data translation/mapping
  - Must address application-level interoperability so that multi-system applications can be described
    - Example: support an installation in a home that contains products from a mixture of systems such as KNX and IGRS, or EchoNet and LonTalk, or Zigbee and HomePlug, etc.
    - Requirement for the interoperability framework to be protocol-independent

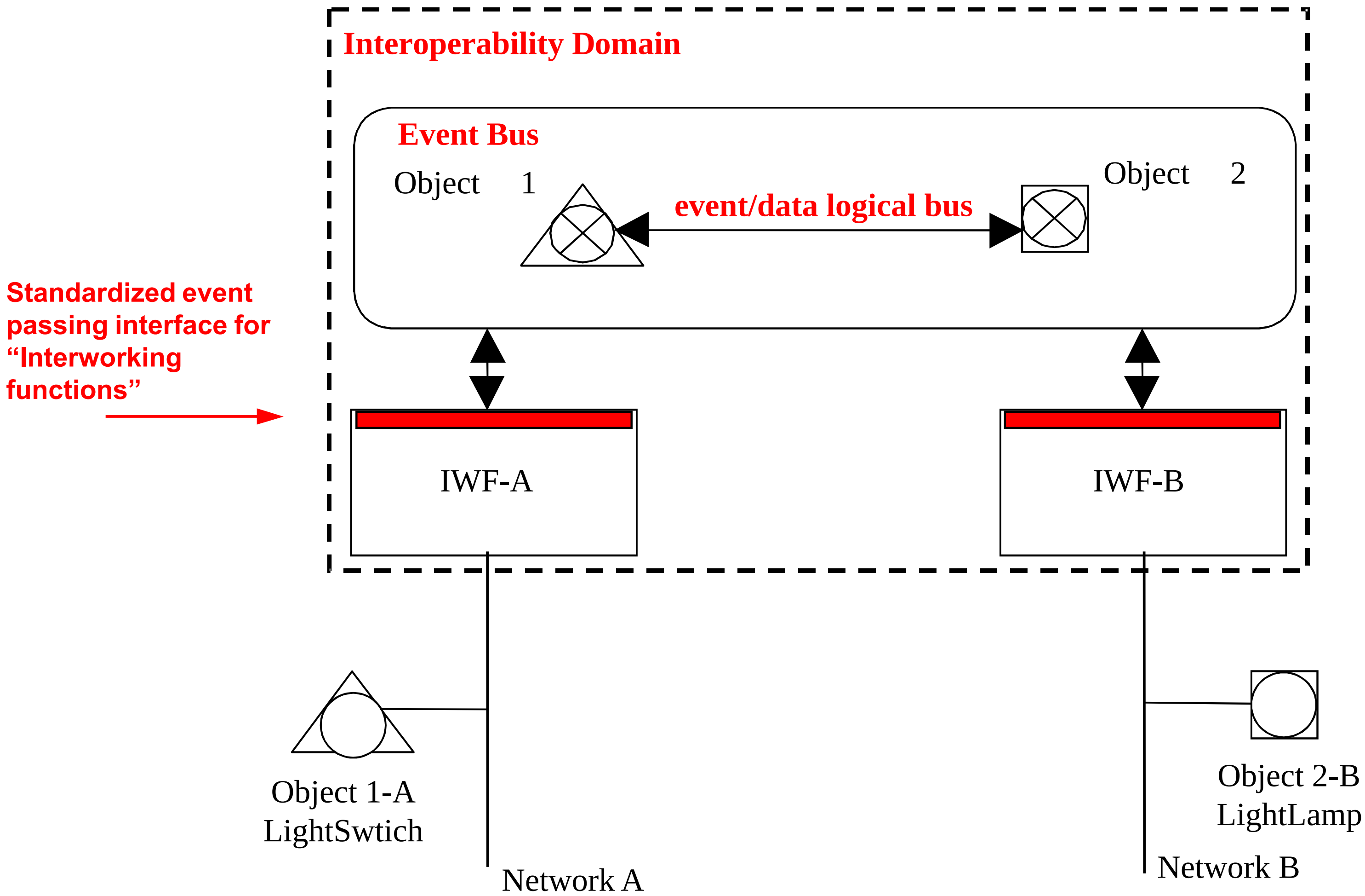
ISO/IEC 18012-2 in GWAC Stack Context: DR example

Category	Layer	DR example
Organizational	8: Economic/Regulatory Policy	We defined assumptions about the real-time market model that we wished to test
	7: Business Objectives	Defer capital investment; improve response to unplanned outages
	6: Business Procedures	Real-time market and buy/sell bids as the primary optimization mechanism
Informational	5: Business Context	Real-time Pricing accounts with customers
	4: Semantic Understanding	***Defined virtual devices that combined the physical device functions with addl business process information flow & functions***
Technical	3: Syntactic Interoperability	***Used an implementation of ISO/IEC 18012-2 to establish heterogeneous interoperability and enable semantic model above***
	2: Network Interoperability	IP and non-IP bridged by a gateway – little or no application function in the gateway
	1: Basic Connectivity	Heterogeneous mix of wired and wireless technologies

HS InteropSys vs. OSI RM - CONCEPTUAL



Application-level Interoperability



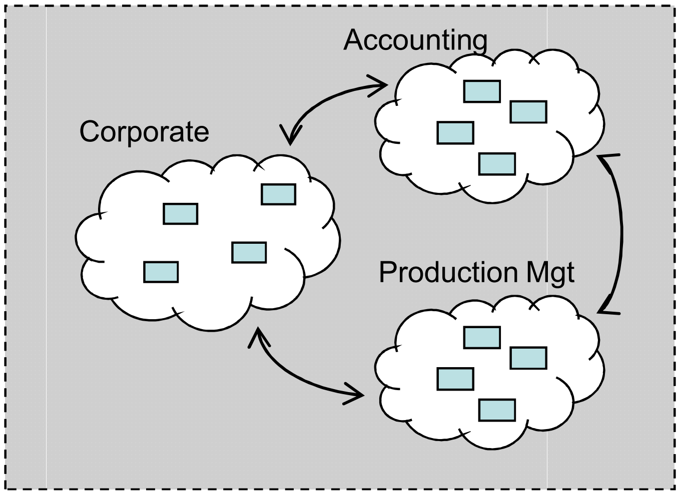
Application-level Interoperability

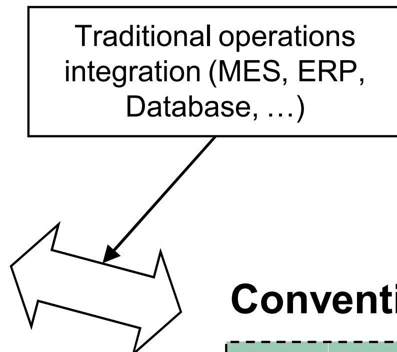
- Basic approach is to describe the interaction between products in a multi-system installation
- Must capture enough information to enable an implementation of the interoperability framework to automatically determine which product parameters need to be transported across the interoperability event bus as a single unit – i.e., to assemble the event message payloads dynamically
  - This effectively breaks the tight association between individual product APIs and their associated protocol definitions for various product functions

AN EVENT-BASED PROGRAMMING, DISTRIBUTED CONTROL SYSTEM VIEW OF INTEROPERABILITY

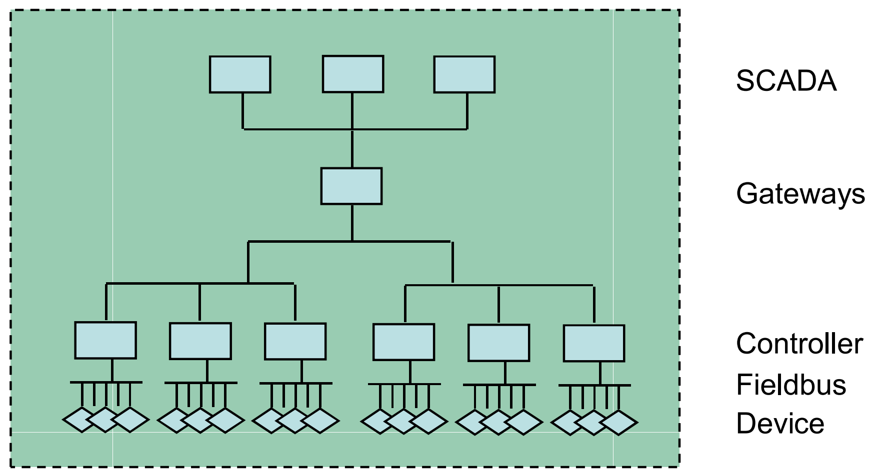
Operational Integration Today

Conventional Enterprise Computing





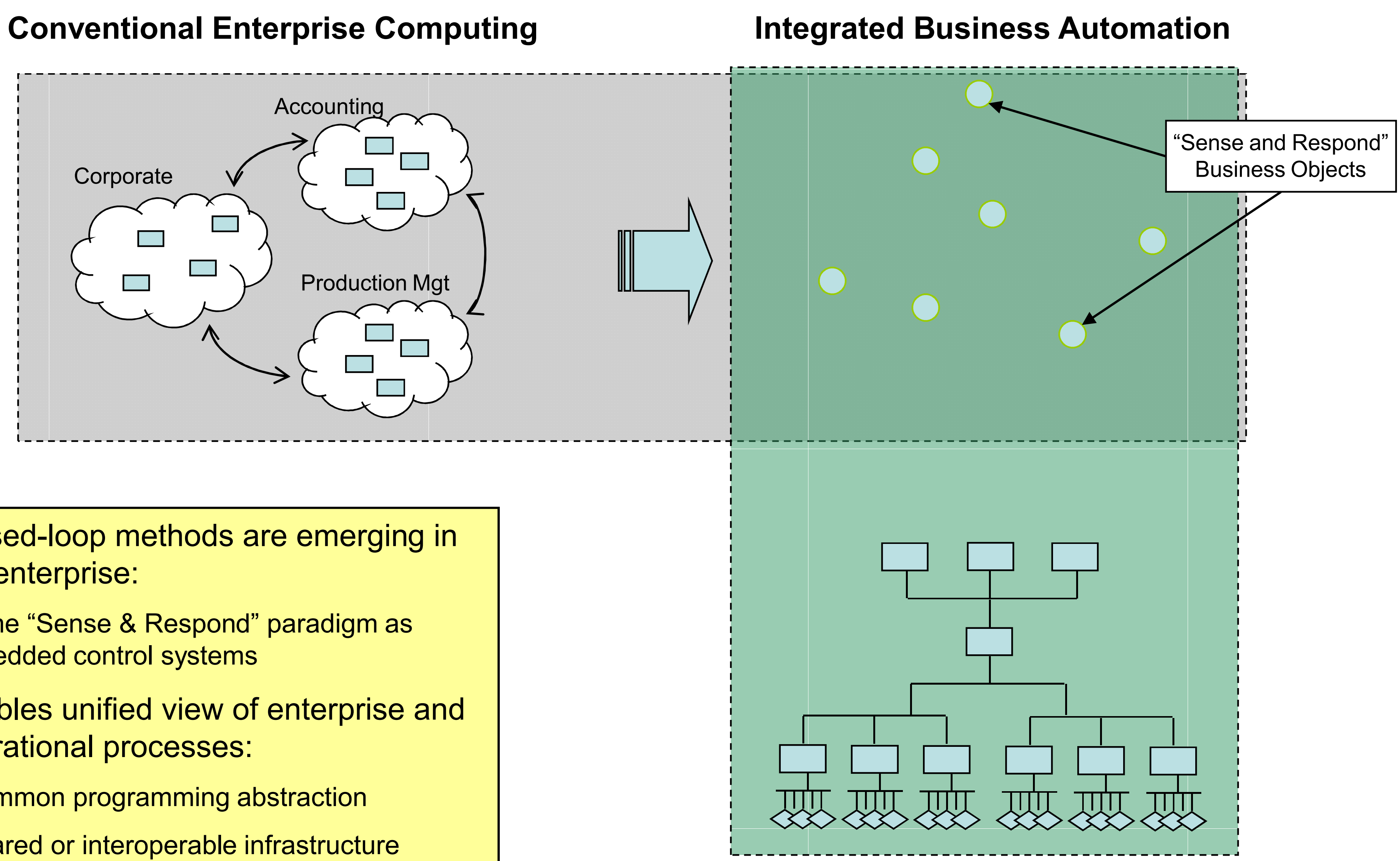
Conventional Operational Control



Today's environment:

- Enterprise & embedded control domains are loosely coupled
- Different programming models

Create Business Value through Operational Integration



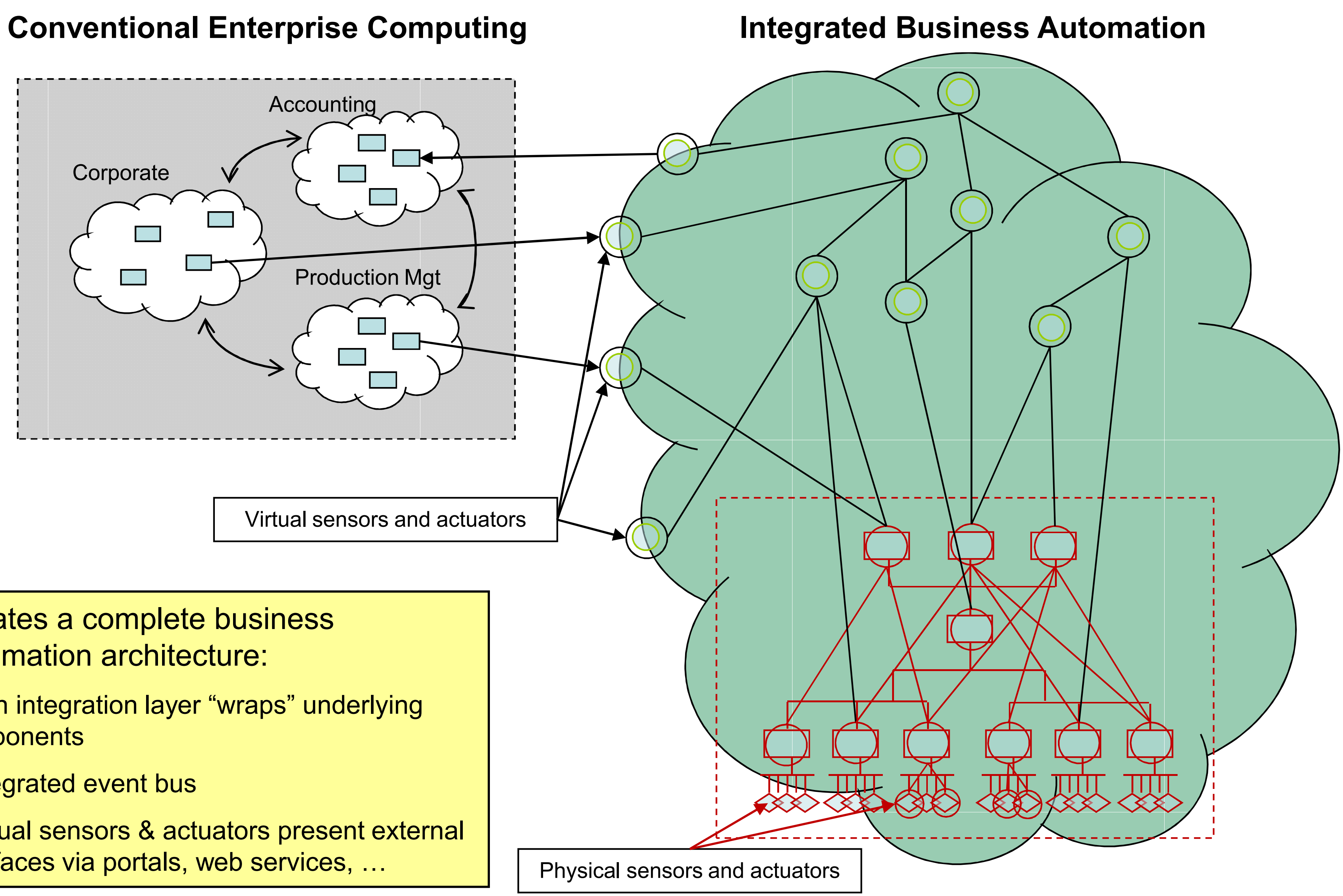
Closed-loop methods are emerging in the enterprise:

•Same "Sense & Respond" paradigm as embedded control systems

Enables unified view of enterprise and operational processes:

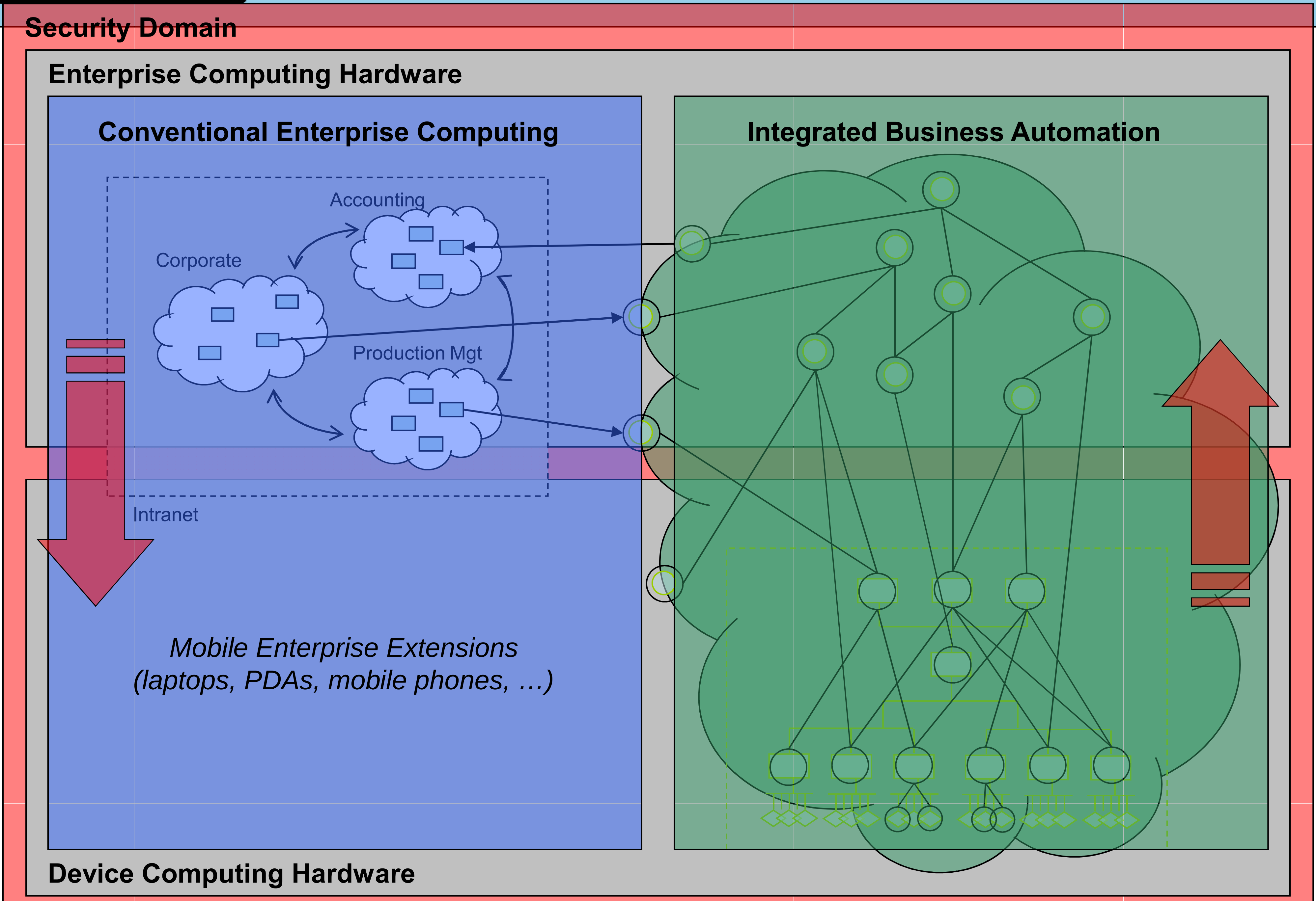
- Common programming abstraction
- Shared or interoperable infrastructure

Establish a Flexible Integration Model based on a logical Event Bus



Creates a complete business automation architecture:

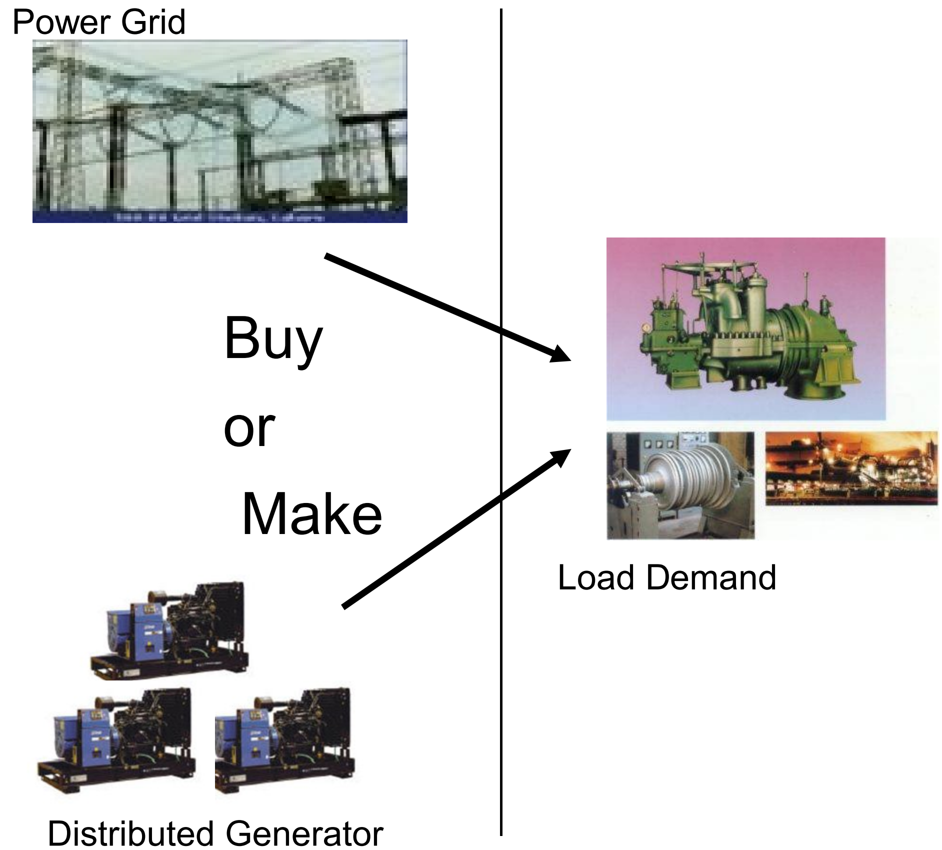
- Thin integration layer "wraps" underlying components
- Integrated event bus
- Virtual sensors & actuators present external interfaces via portals, web services, …


Security Domain
Enterprise Computing Hardware
Conventional Enterprise Computing
Accounting
Corporate
Production Mgt
Intranet
Mobile Enterprise Extensions
(laptops, PDAs, mobile phones, …)
Integrated Business Automation
Device Computing Hardware

SIMPLE ELECTRIC ENERGY INDUSTRY EXAMPLE

Simple Distributed Energy Management example

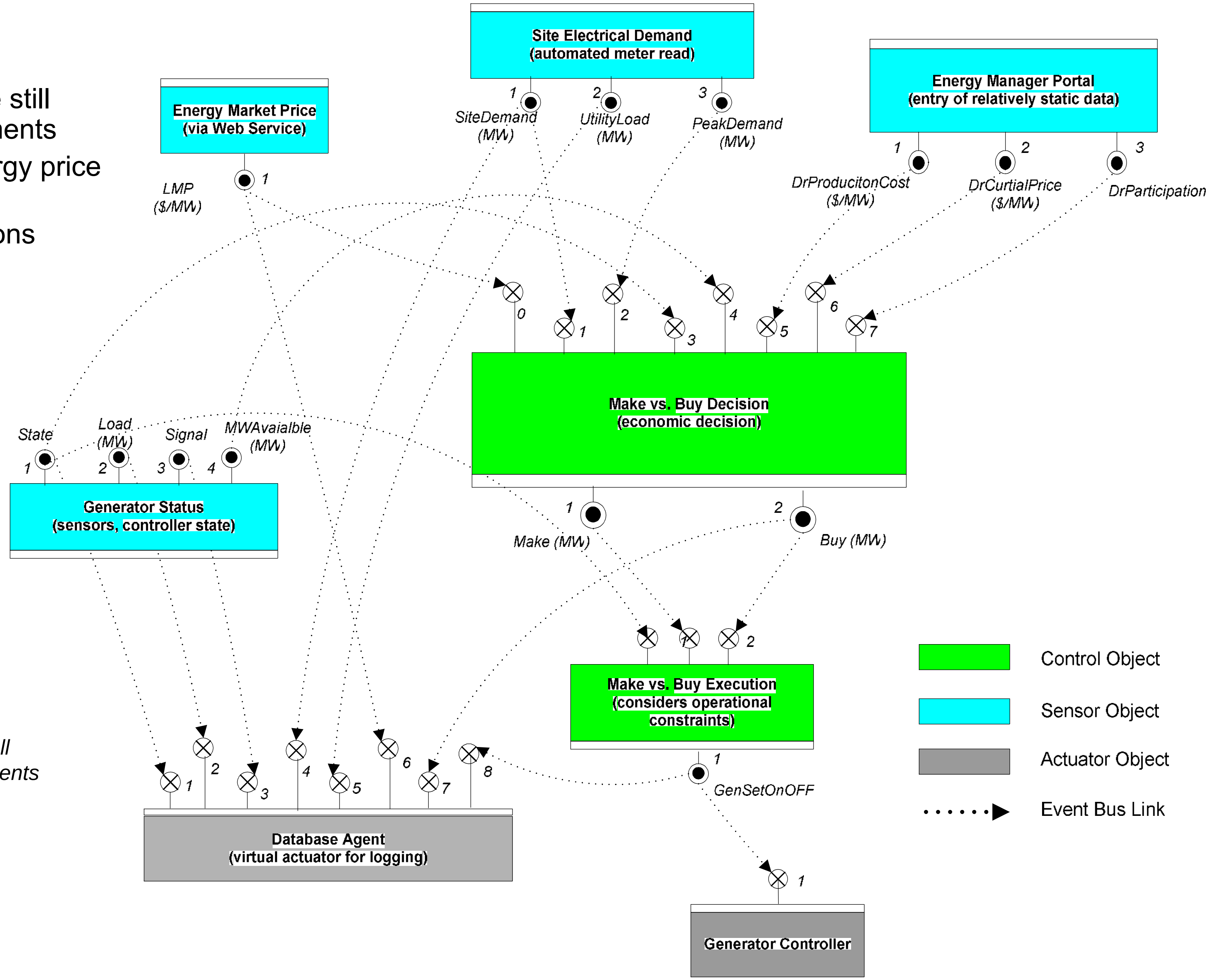
- Local electrical power make/buy decision:
  - Cost of running local generator vs. cost of buying power from electrical grid.
  - Three local generators with different capacity and different energy-conversion efficiency (kwh/gallon-fuel) are under control.
  - The overall cost of running generator to kwh electricity power depends on price of fuel, efficiency of generator, licensed-hour to run, operational cost.
  - Once the decision of local generation is lower cost than buying from grid, one or more generators are turned on depends on current demand.
  - At given time, the local site may be:
    - - Completely powered by electrical grid
    - - Completely powered by local generator (s)
    - - Partially powered by both gird and local generator (s)



Simple Distributed Energy Management example

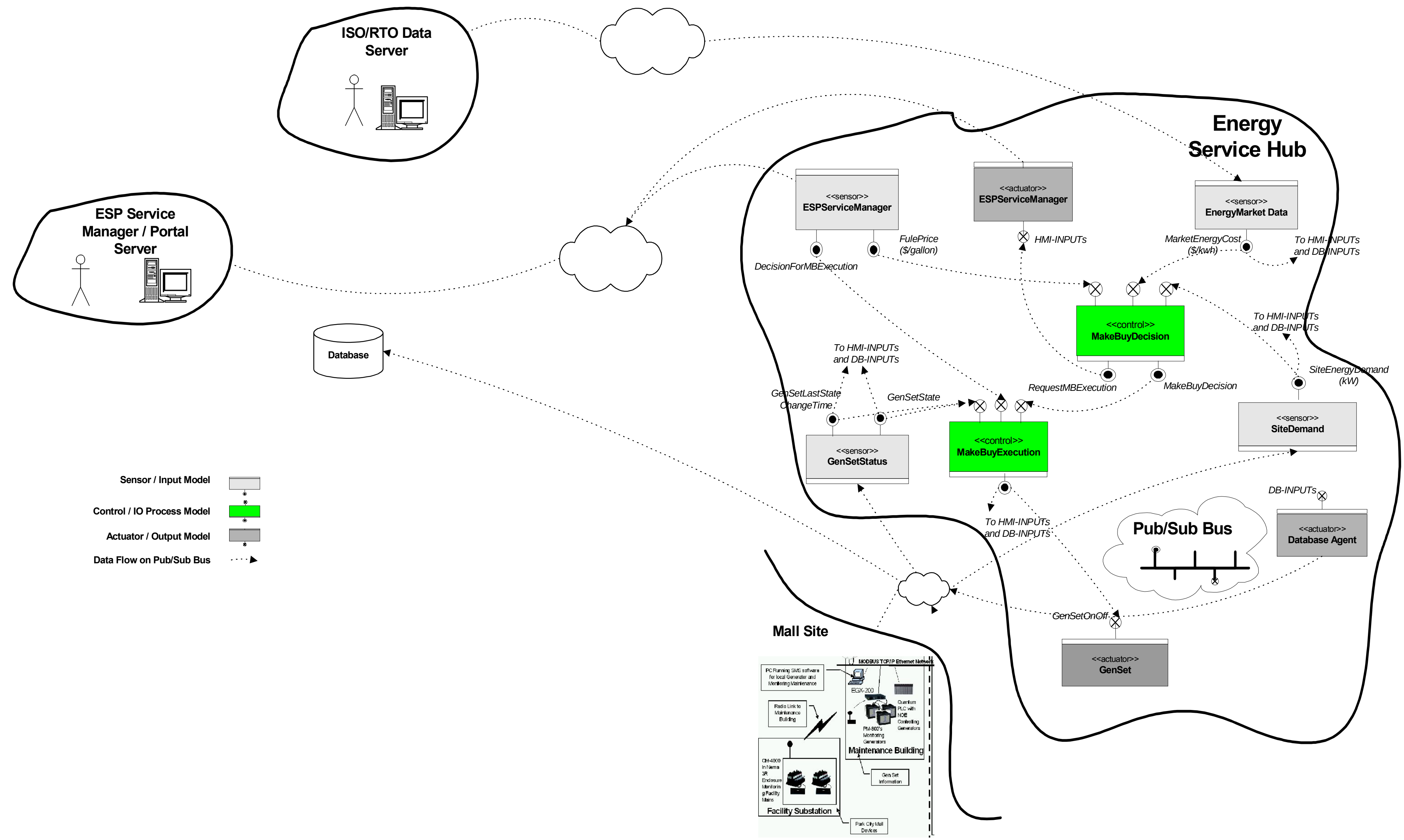
Objectives …

- Minimize energy cost while still meeting business requirements
- Protect business from energy price spikes
- Monitor & manage operations remotely

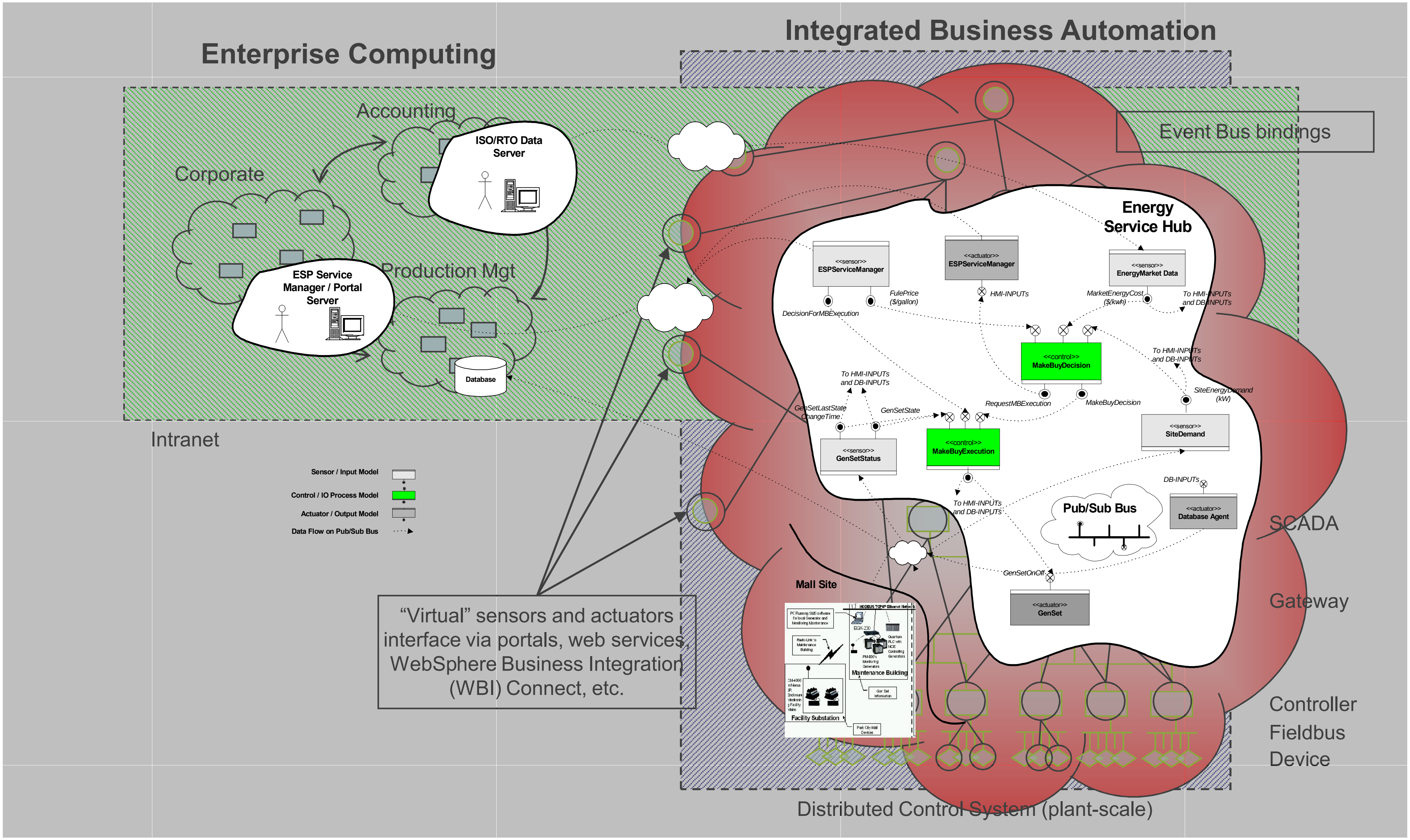


Note: *All application objects, and overall binding graph, declared by XML documents based on defined schema*

Simple Distributed Energy Management example



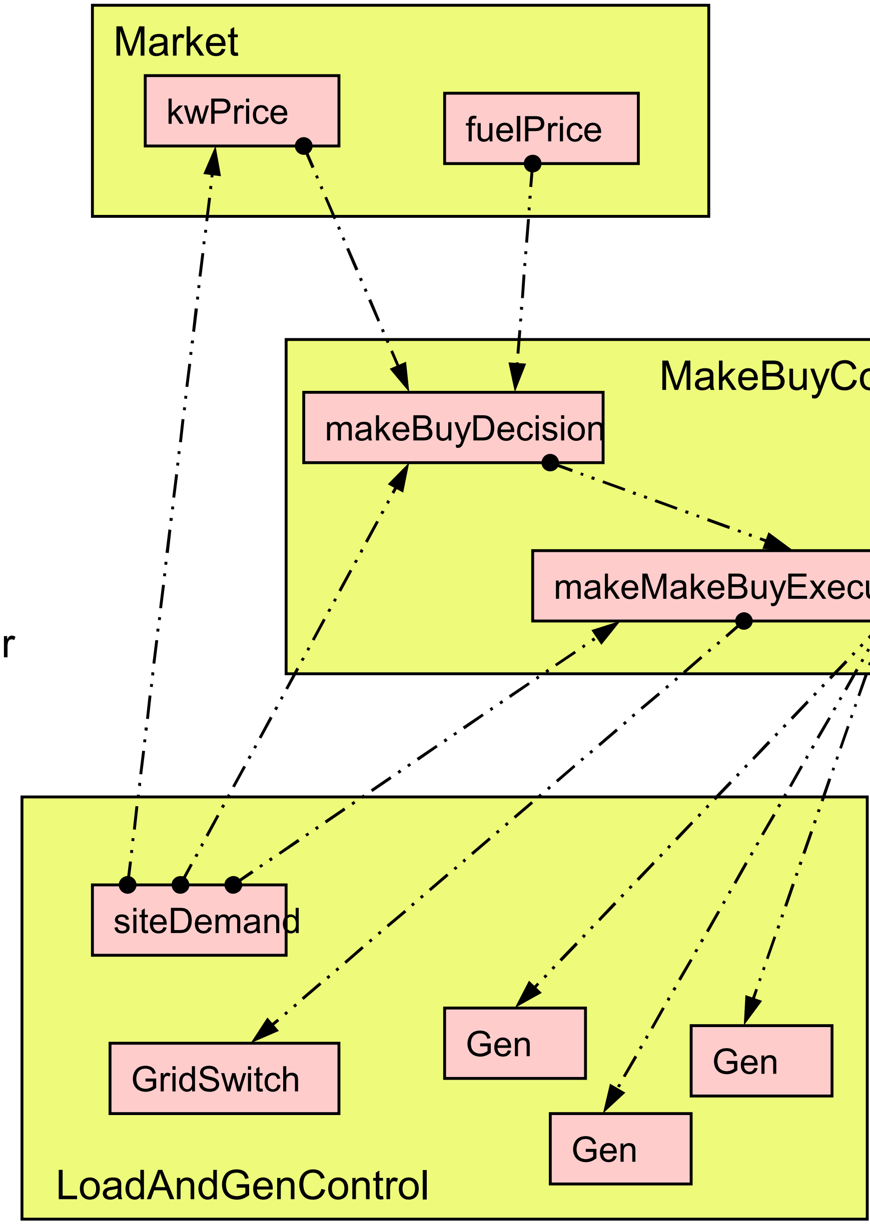
Simple Distributed Energy Management example



Simple Distributed Energy Management example

- Key Control Element
  - kwPrice(sensor): unit wholesale energy market price
  - fuelPrice(sensor): unit price of fuel for generator
  - makeBuyDecision(controller): decision algorithm for make/buy
  - maekBuyExecution(controller): decision algorithm for running the generator and grid switching
  - siteDemand(sensor): demand metering for site
  - GridSwitch(actuator):
  - Gen(actuator): generator controller

- Three iCS Runtime Nodes
  - Market: market information source
  - MakeBuyControl: decsion/control of make/buy
  - LoadAndGenControl: generator control and grid switching



A prototype implementation of ISO/IEC 18012

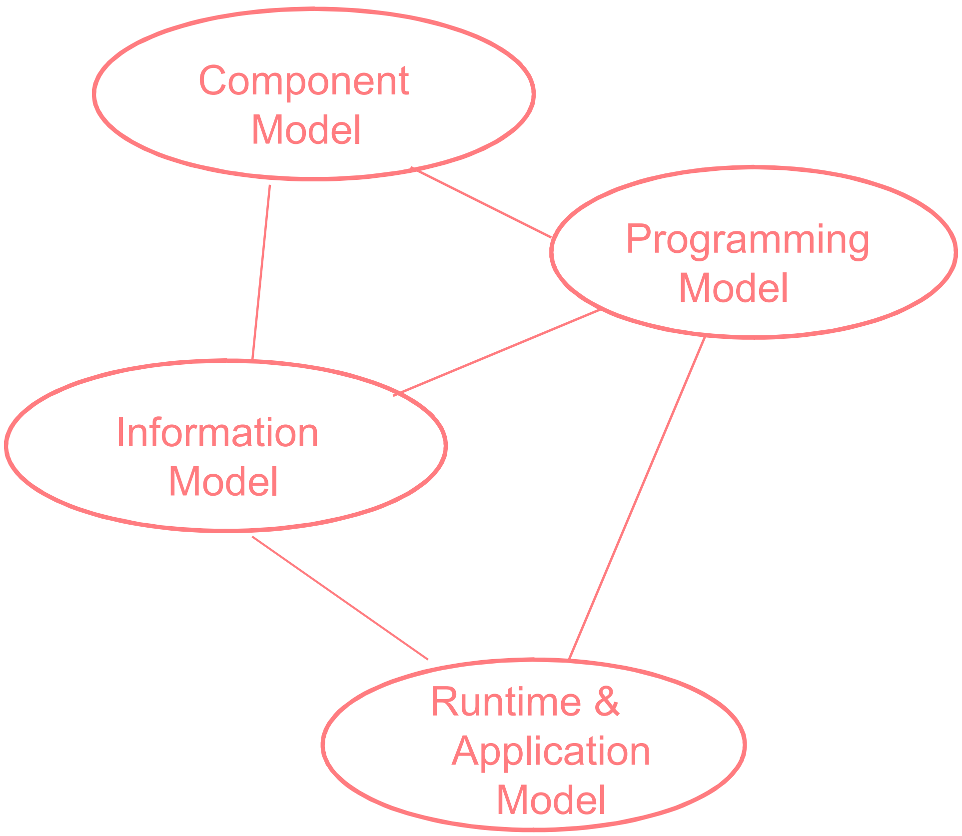
INTERNET-SCALE CONTROL SYSTEMS (ICS) DESIGN AND IMPLEMENTATION

Internet Control System (iCS)

- Model business components and processes as control elements and decision-loops
- Virtualization of physical sensor/actuator/devices with object/component, model-driven approach
- Introduce Middleware/Application Service To Physical Control Domain
- Scalability from embedded control to enterprise server environment
- Hybrid Application Model:
  - Data/Event Messaging (pub/sub, point-to-point), Event Correlation
  - Service/Function Invocation, Request/Response
- Separation of application logic (software components and links) from computational infrastructure (system hardware and network topology)
- Separation of concerns
  - Application domain expert, function block/module builder
  - Application Integrator
  - Application User/Administrator

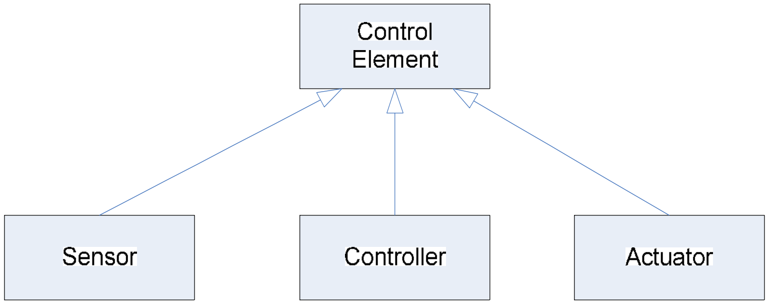
Model Driven Architect Approach

- Component Model
  - Structure of Control Element
  - Composition and application logic graph
- Programming Model
  - Continuous decision control-loop vs. transaction flow
  - Event/Messaging vs. Service Choreography
  - Separation of concerns
- Information Model
  - Types of data, event, IO at both physical, virtual and programming levels
  - XML Schema
- Runtime & Application Model
  - Communication & Interaction Model
    - How Control Elements interact and communicate with each other in memory or cross network
  - Physical Distribution Model
    - How Control Elements are distributed on networked computation nodes



Component Model: Control Element (iCS Building Block)

- All application components in iCS are modeled as control element, and there are three types of control elements:
  - Sensor: data/event source, output only
  - Controller: data/event processor, input and output
  - Actuator: data/event sink, input only

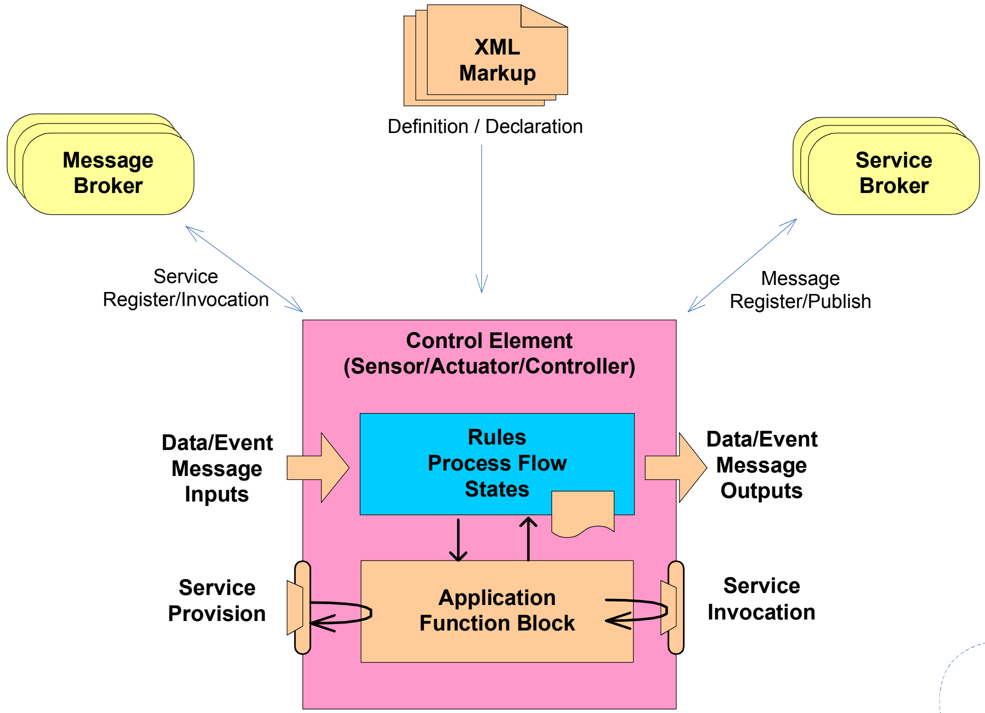


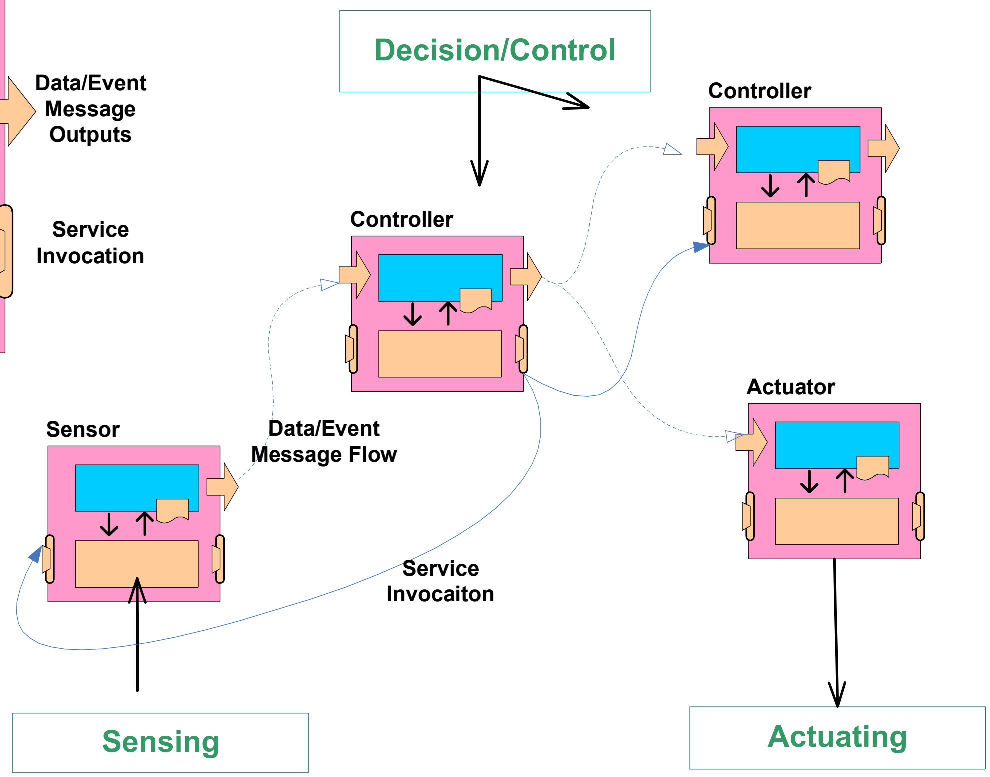
  - All three types of control elements can also be modeled
  - with service interfaces as service consumer and provider

- Advantages:
  - Simplified application component model that preserve the characteristics of control system model, yet can be used model most business information components and processes
  - Well suited for event messaging programming model
  - Well suited for adapter and containment design pattern for integration of heterogeneous components and other programming models
  - Simple enough to be easily and fully described with XML information model

Programming Model

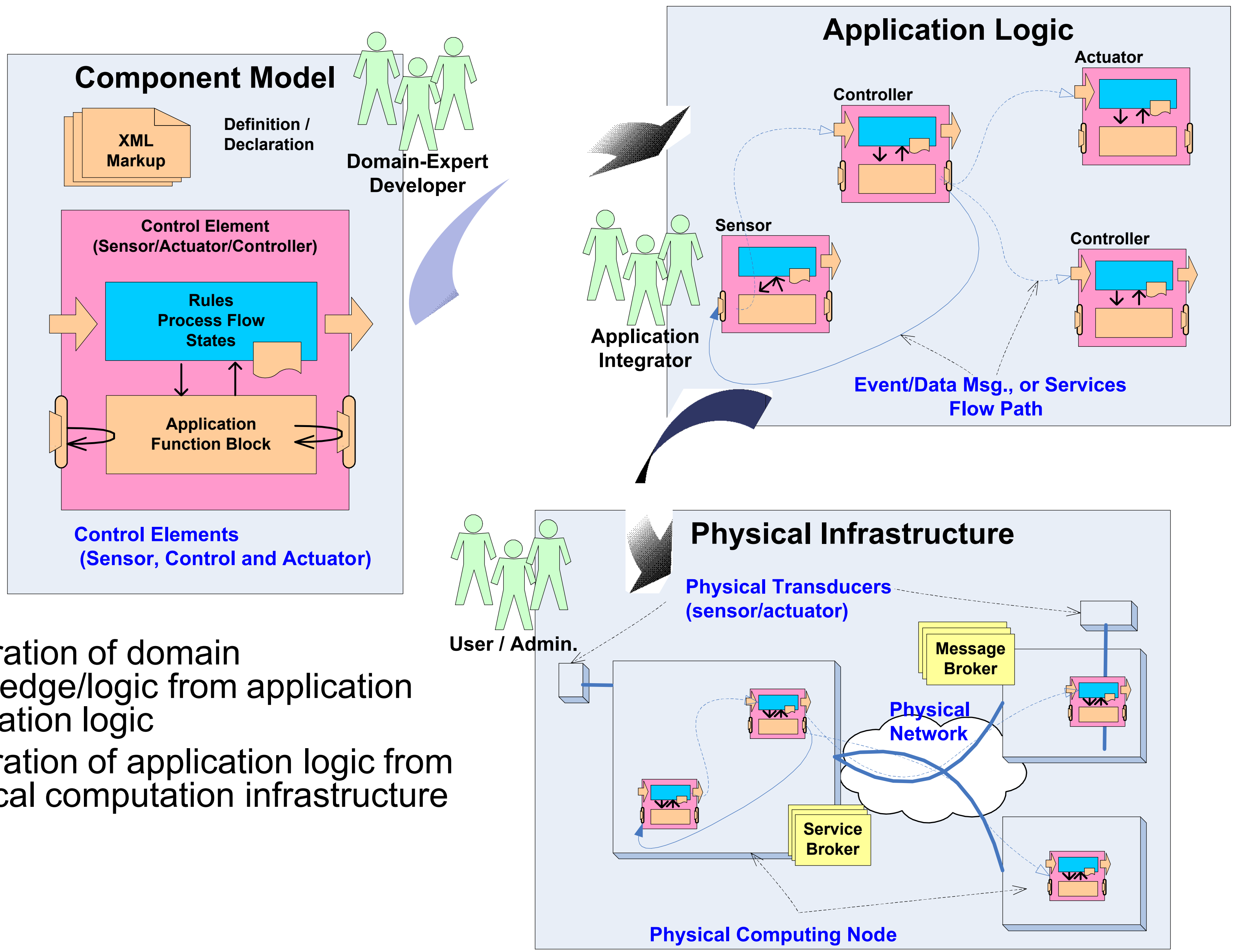
- Hybrid Programming Interfaces for control element: Data/Event Message and Service Function Interface
- Data/event message interaction is achieved using Message-Broker network (pub/sub, point-2-point)
- Service function interaction is provided by Service Broker





- Application Logic is a combination of
  -Path routing of data/event message among a set of control elements
  -Choreography of service invocation flow of one or more of control elements

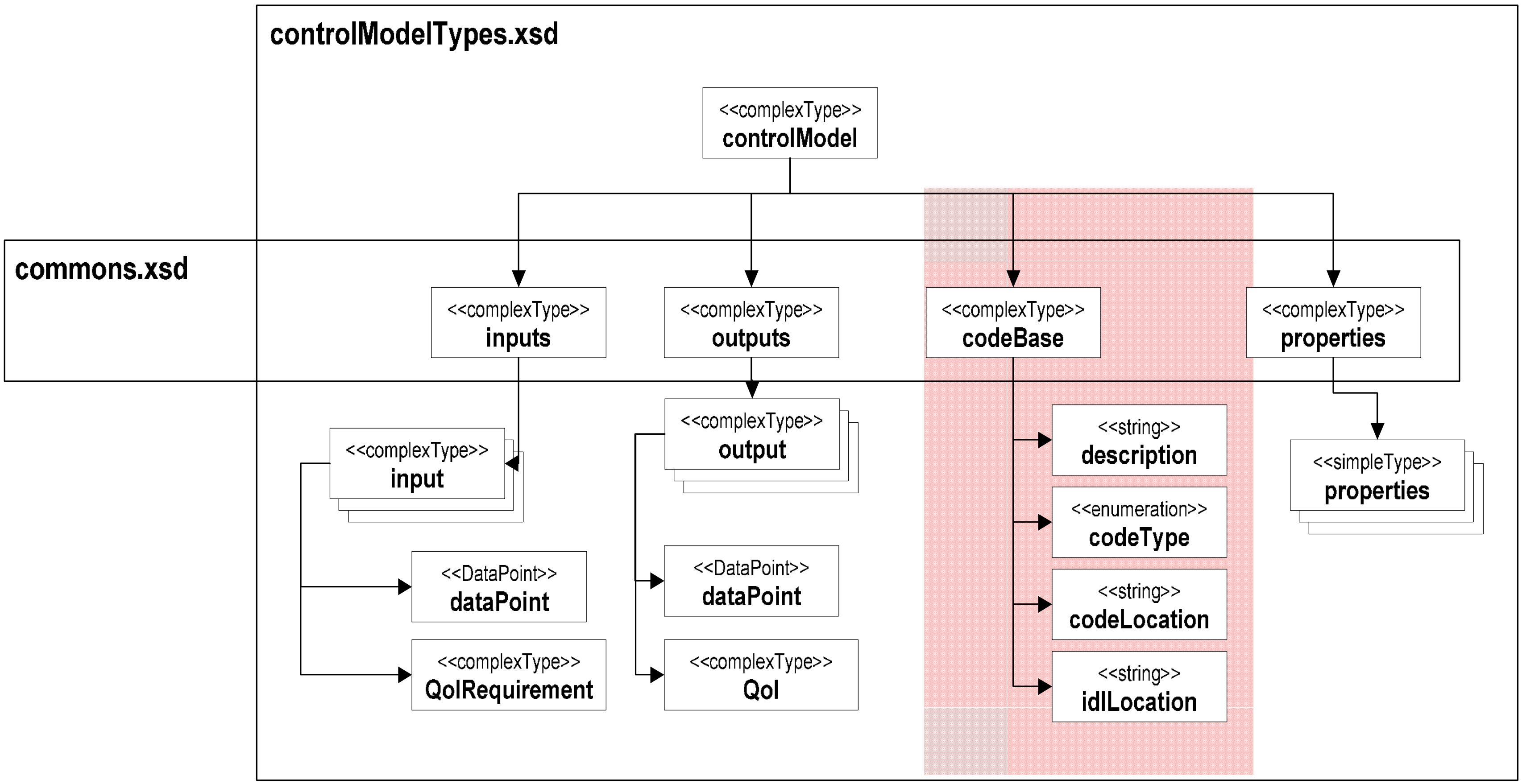
Programming Model: Separation of Concerns



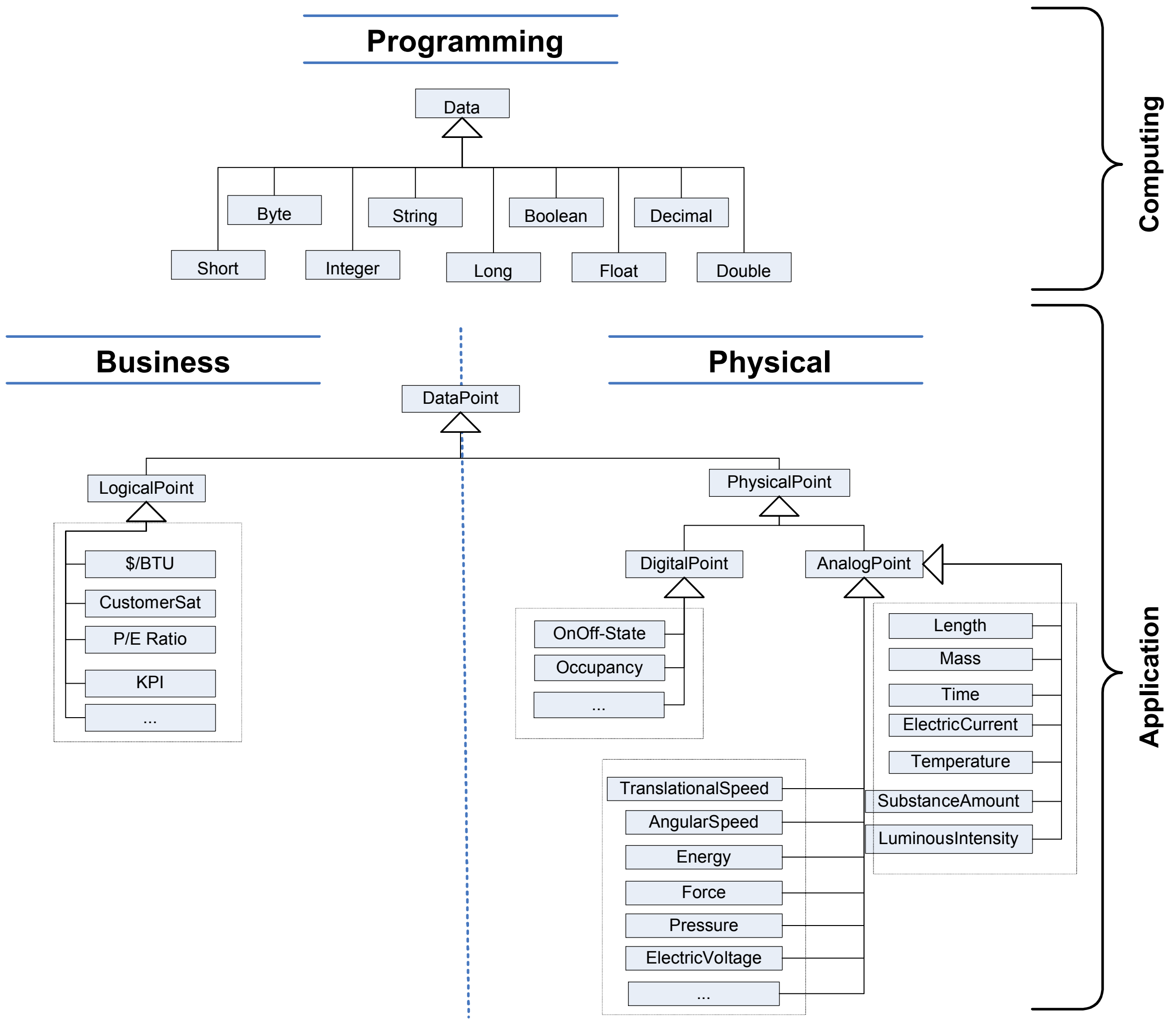
- Separation of domain knowledge/logic from application integration logic
- Separation of application logic from physical computation infrastructure

Information Model: Control Element XML Schema

- All Control Element are defined and declared with XML
  - Input/Output: type, unit of physical quantity, uncertainty, frequency, etc.
  - Model Properties: data/event correlation rules, algorithm trigger rules, execution threading model
  - Code Base: application-code reference, application-code specific properties, parameters

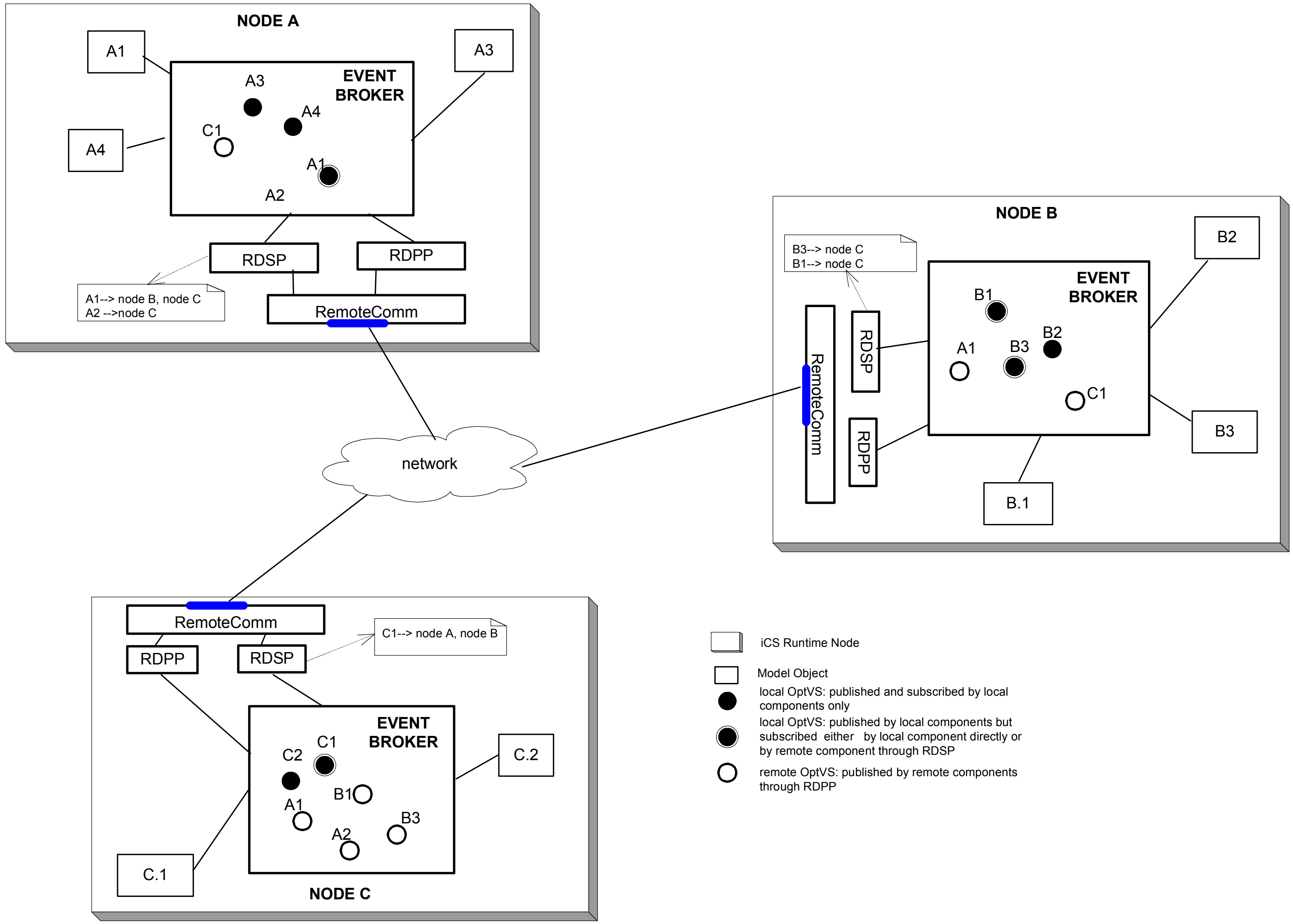


XML Schema: Data Point Types and Physical Units

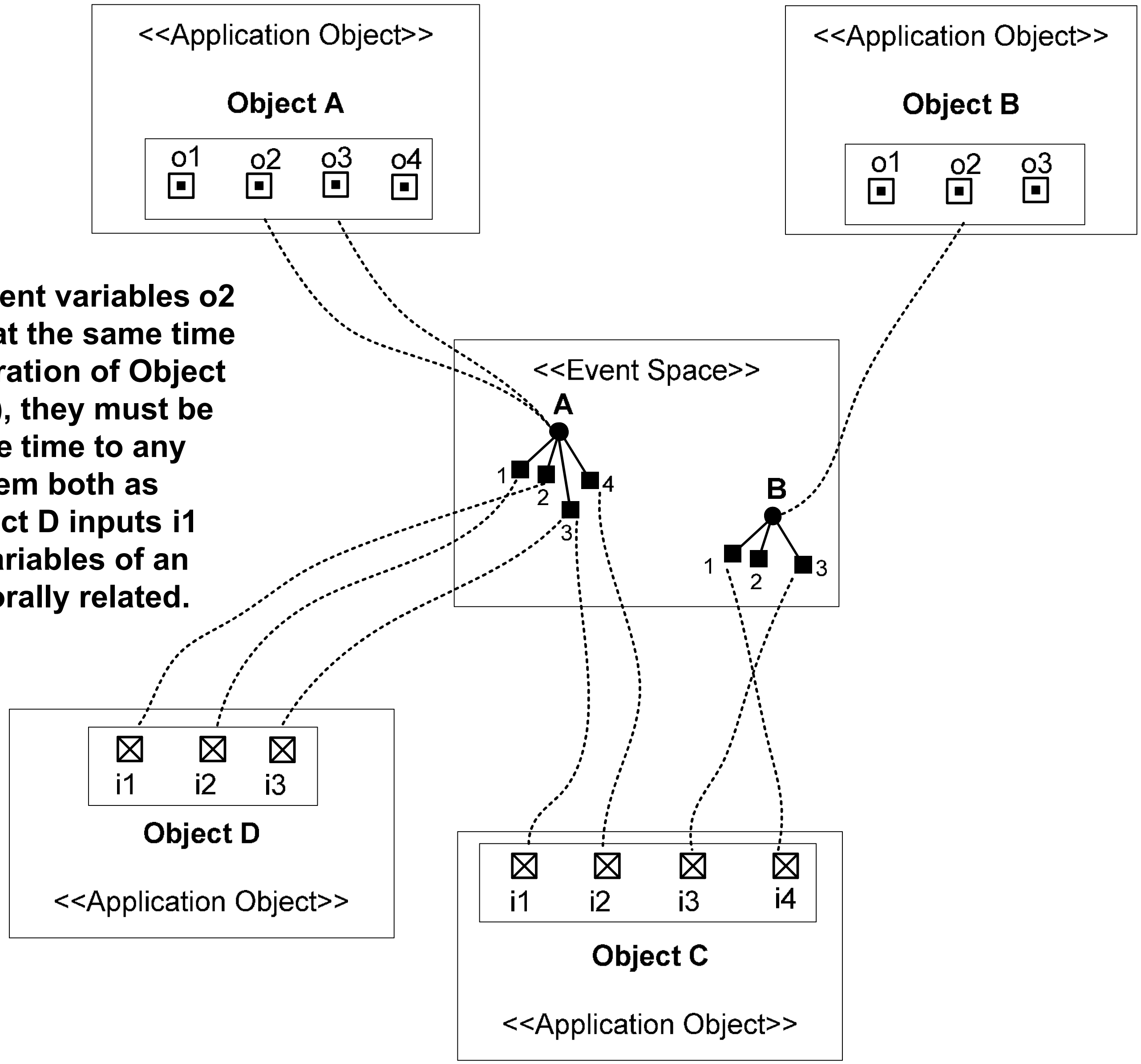


- Data type framework to map to computing space from business and operations domains
- Data types are extensible through XML schema to different industries and business domain
- Provide data type platform for Model Object Inputs/Outputs

Distributed Event Space Implements a Logical Event Bus

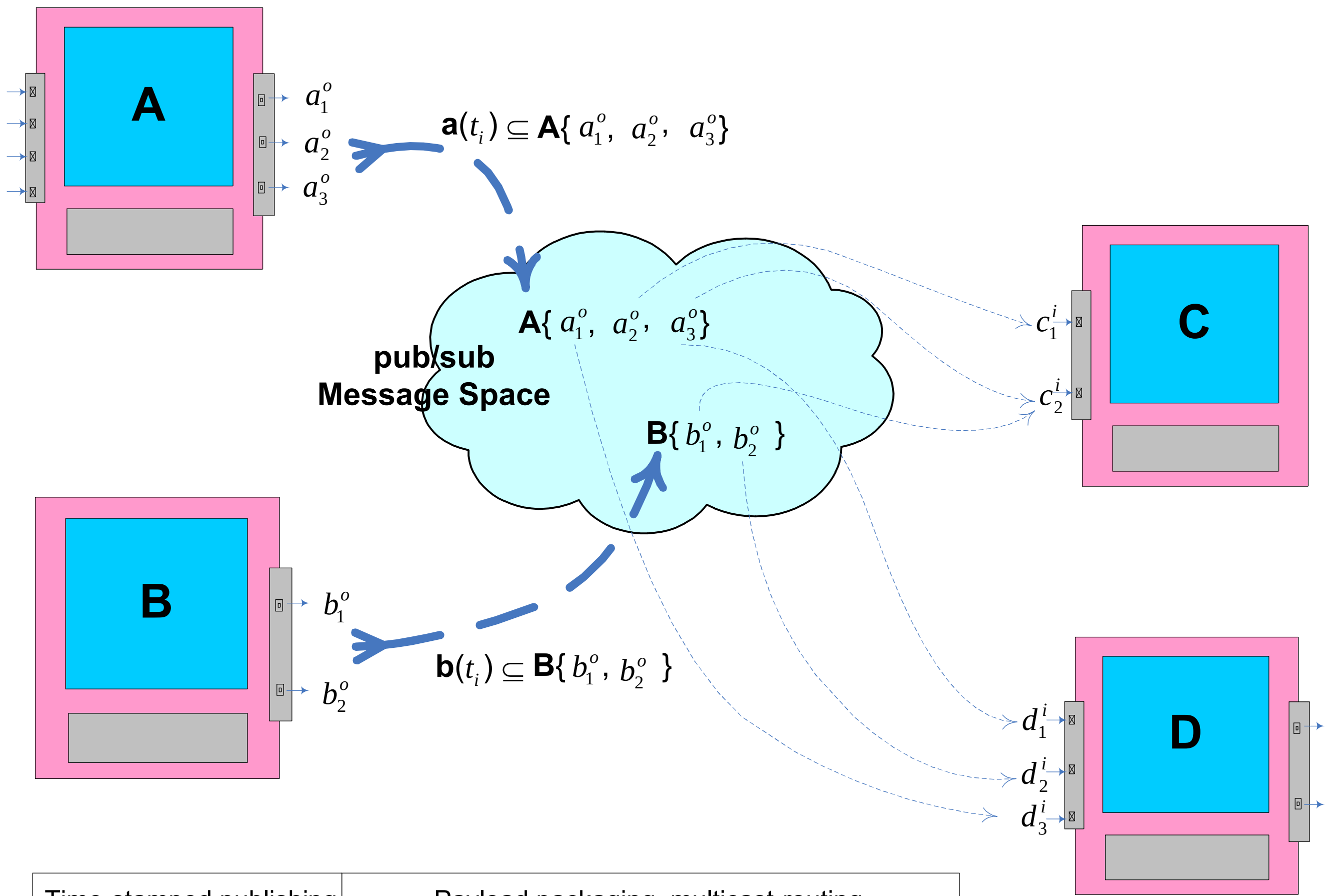


Event temporal-relationships are maintained



If Object A output event variables o2 and o3 are updated at the same time (i.e., on the same iteration of Object A's model algorithm), they must be delivered at the same time to any Objects that have them both as inputs, such as Object D inputs i1 and i3 – all output variables of an iCS object are temporally related.

Communication Model: mapping of I/O events from source to receiver (pub/sub or p-2-p)

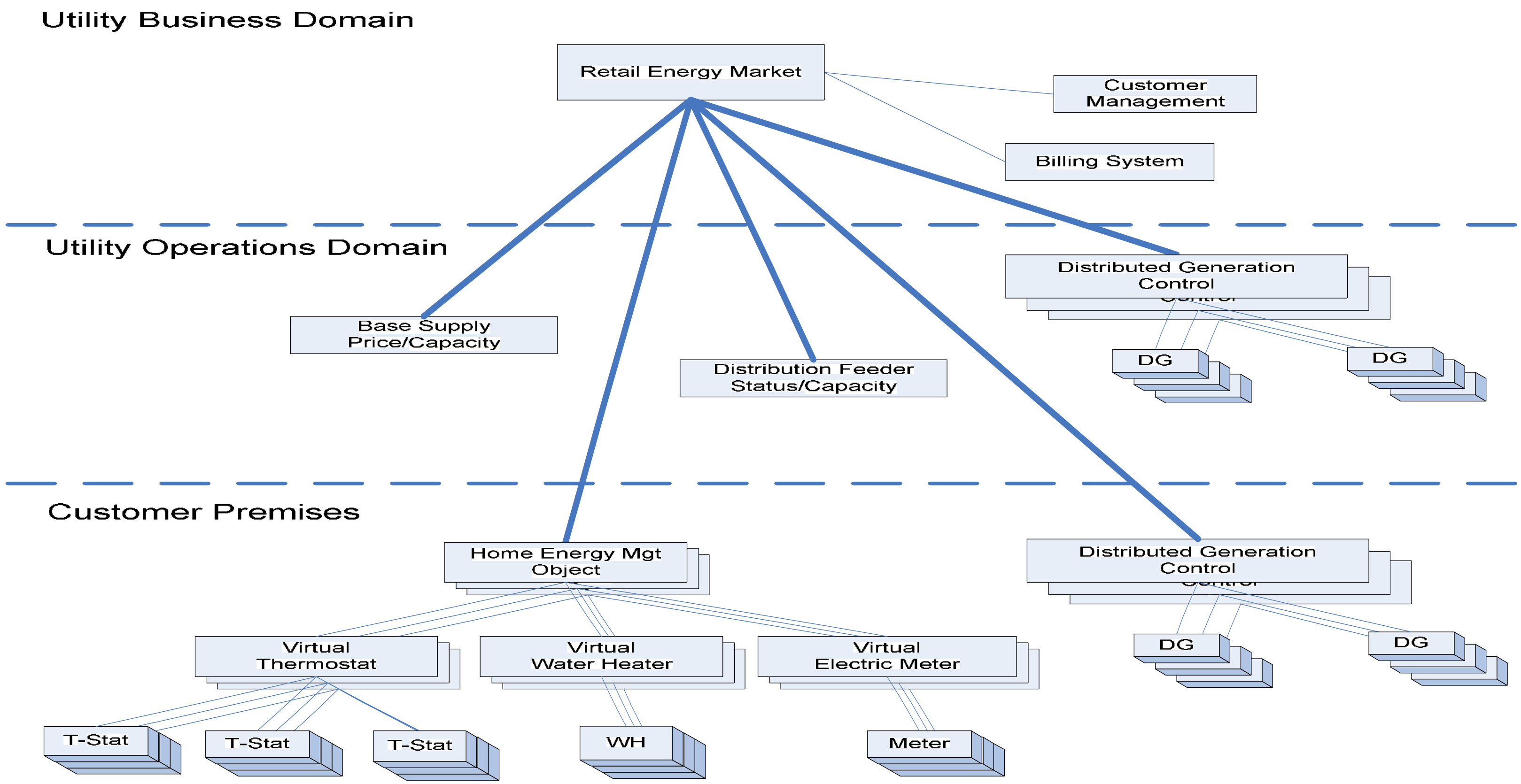


- Message I/O mapping, routing
- Message payload packaging
- Temporal Association

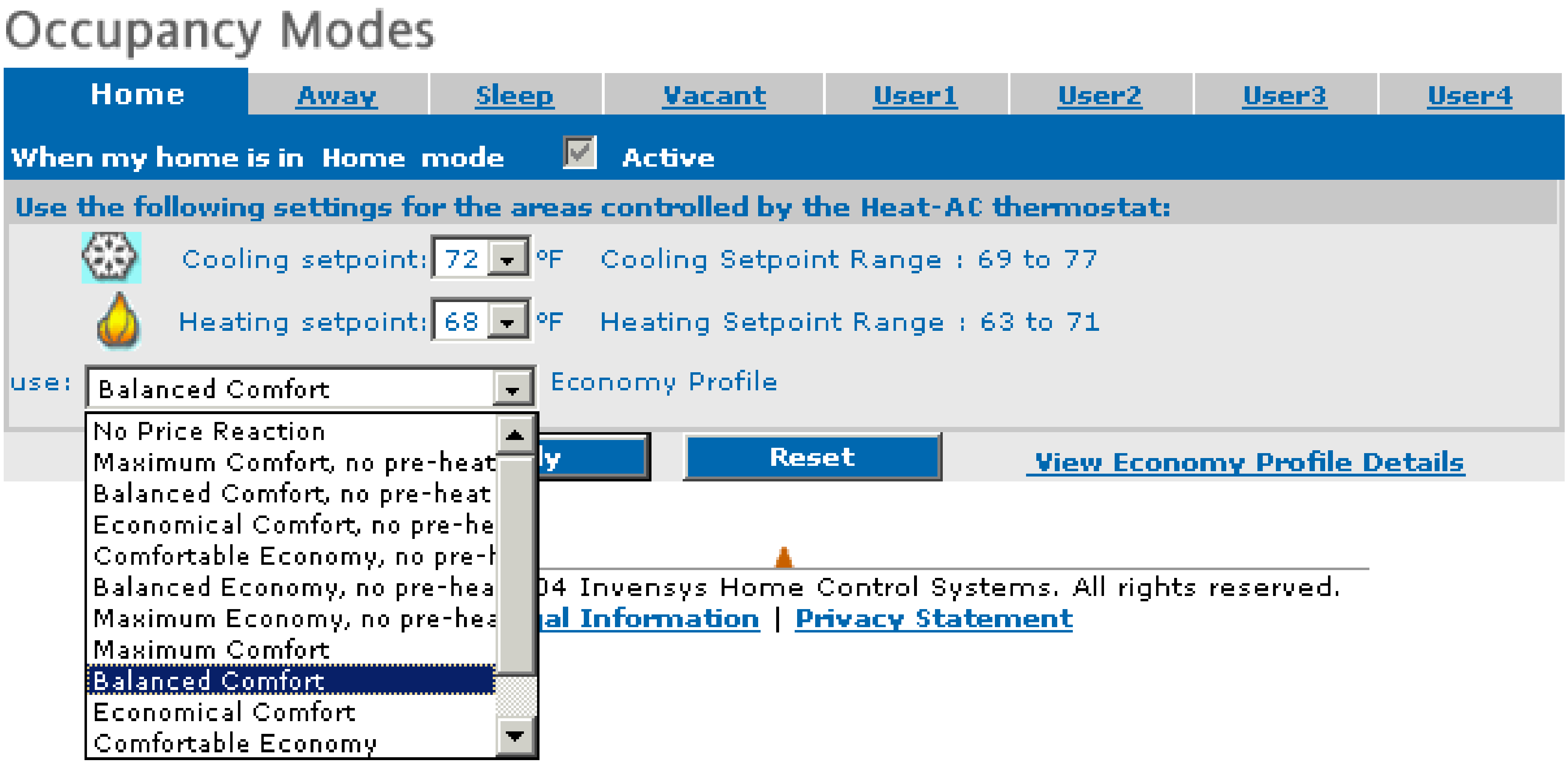
Time-stamped publishing		Payload packaging, multicast-routing	
	sender **A,B**	Receiver **C**	Receiver **D**
t_1	$\mathbf{a}\{a_2^o\}$	$\mathbf{c}\{c_1^i\}$	null
t_2	$\mathbf{a}\{a_1^o, a_2^o, a_3^o\}$	$\mathbf{c}\{c_1^i, c_2^i\}$	$\mathbf{d}\{d_1^i, d_3^i\}$
t_3	$\mathbf{b}\{b_1^o, b_2^o\}$	$\mathbf{c}\{c_2^i\}$	$\mathbf{d}\{d_2^i\}$

OLYMPIC PENINSULA PROJECT VIRTUAL THERMOSTAT EXAMPLE

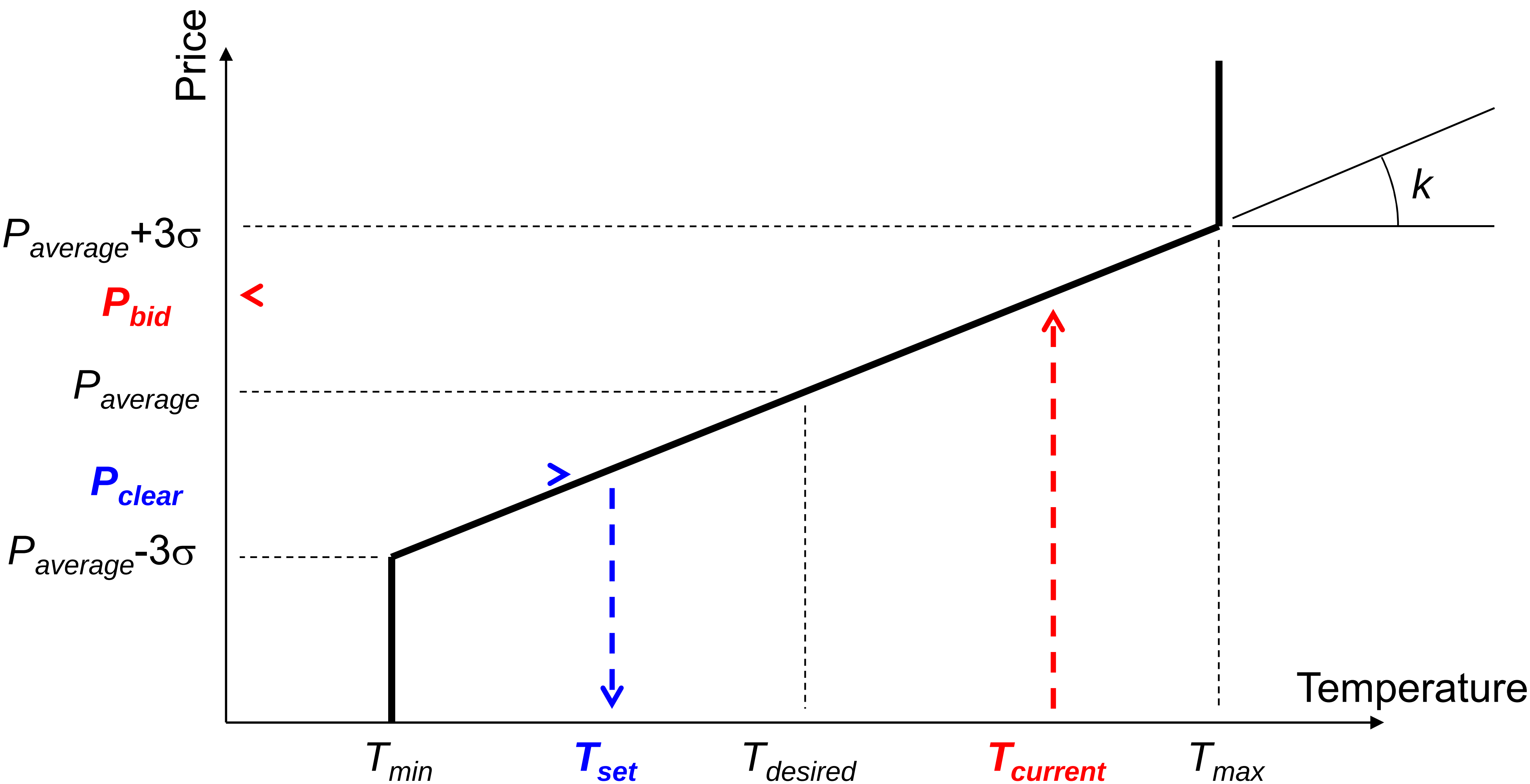
The Virtual Thermostat Object



User Goal-based Preferences



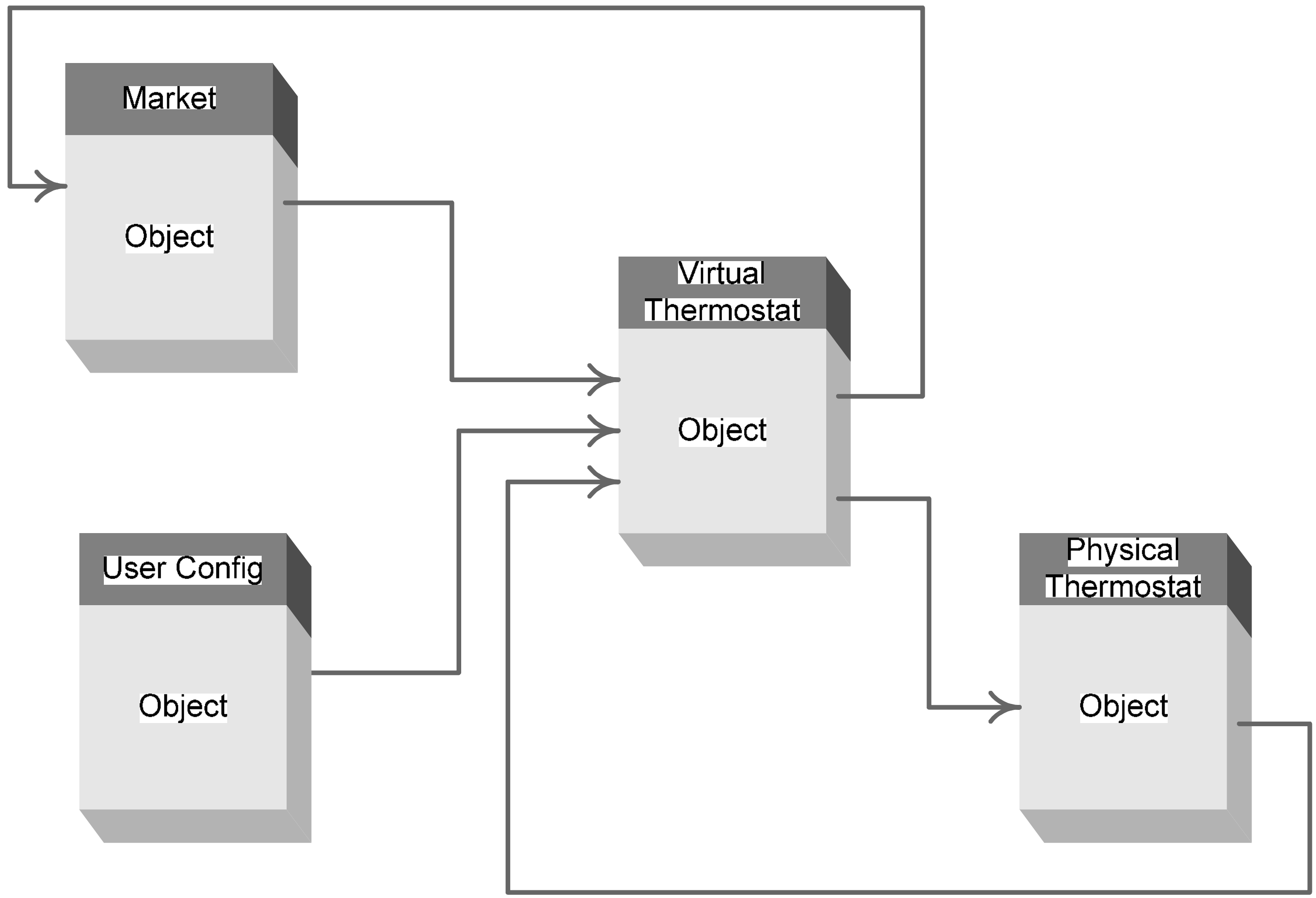
Virtual T-Stat Control Graph



Small *k*: low comfort, high demand response

Large *k*: high comfort, low demand response

Virtual T-Stat Event Connections



Grid-Interop™

Contact

Ron Ambrosio
Global Research Executive
Energy & Utilities Industry

Ron Ambrosio/Watson/IBM@IBMUS
rfa@us.ibm.com

+1 914-945-3121

IBM T.J. Watson Research Center
P.O. Box 218
1101 Kitchawan Rd. / Route 134
Yorktown Heights, NY 10598