TRUST
Team for Research in Ubiquitous Secure Technology

Model-Based Design Environment for Clinical Information Systems

Janos Mathe, Sean Duncavage, Jan Werner,
Akos Ledeczi, Bradley Malin, Janos Sztipanovits

Vanderbilt University

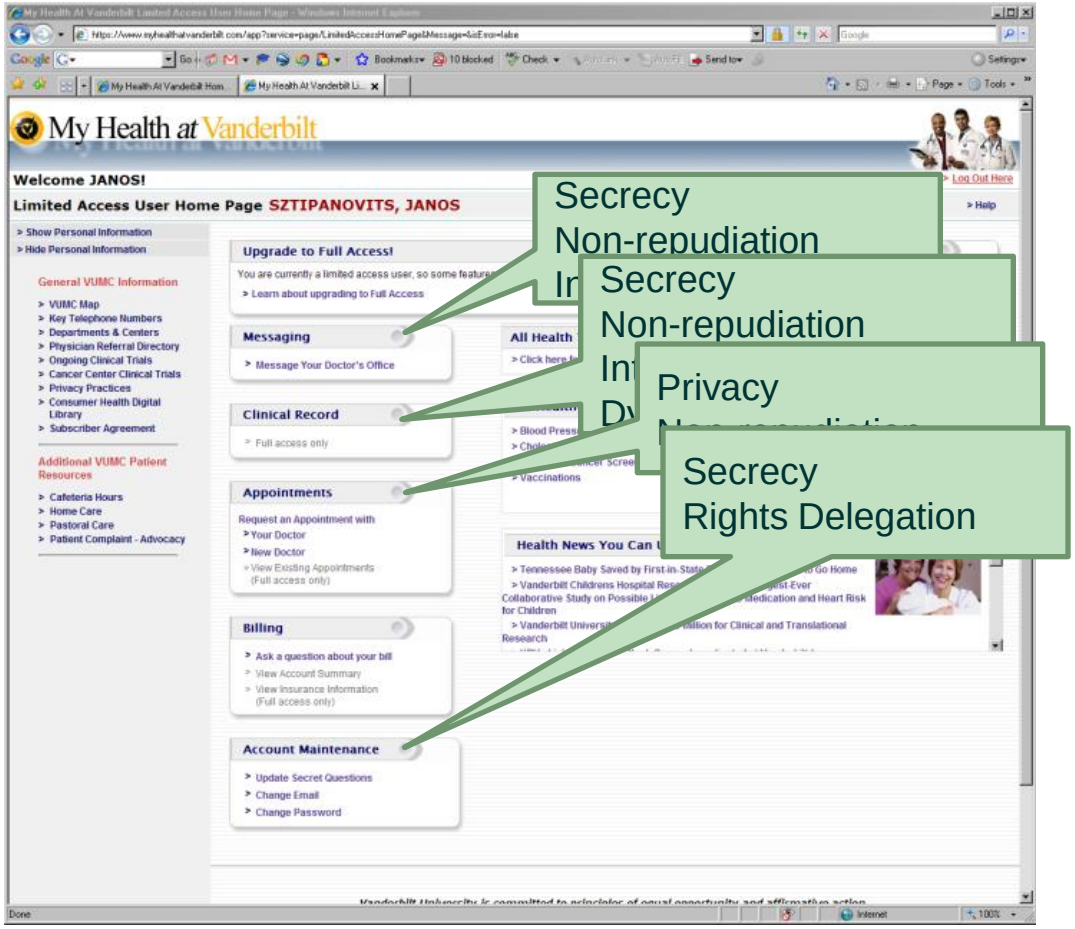
Carnegie Mellon
Cornell University
MILLS COLLEGE
San José State UNIVERSITY
SMITH COLLEGE
STANFORD UNIVERSITY
Berkeley
VANDERBILT UNIVERSITY

TRUST Review, October 11, 2007

Summary

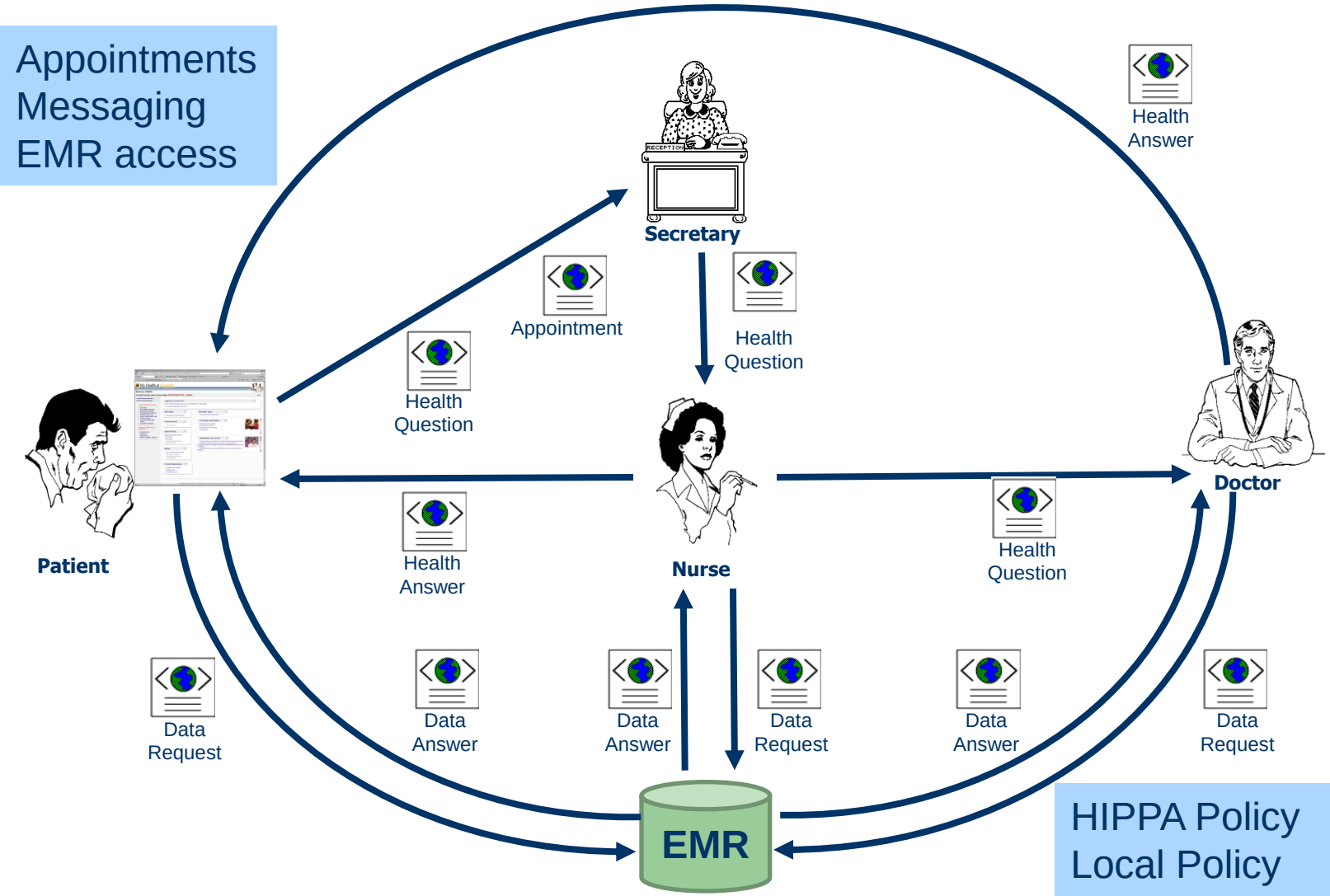
- EMR is an integrative project with three main goals:
  - Build a credible testbed for EMR research
  - Contribute to solving privacy and security challenges of EMR *systems* applications
  - Use EMR application testbeds for the integration, testing and evaluation of new technologies on the following core TRUST research areas:
    - Model-based design for security and privacy
    - Formal modeling, verifying and enforcing privacy and security policies
    - Security and privacy technologies for sensor networks
    - Public policy to technology interactions

Patient Portal Research Area



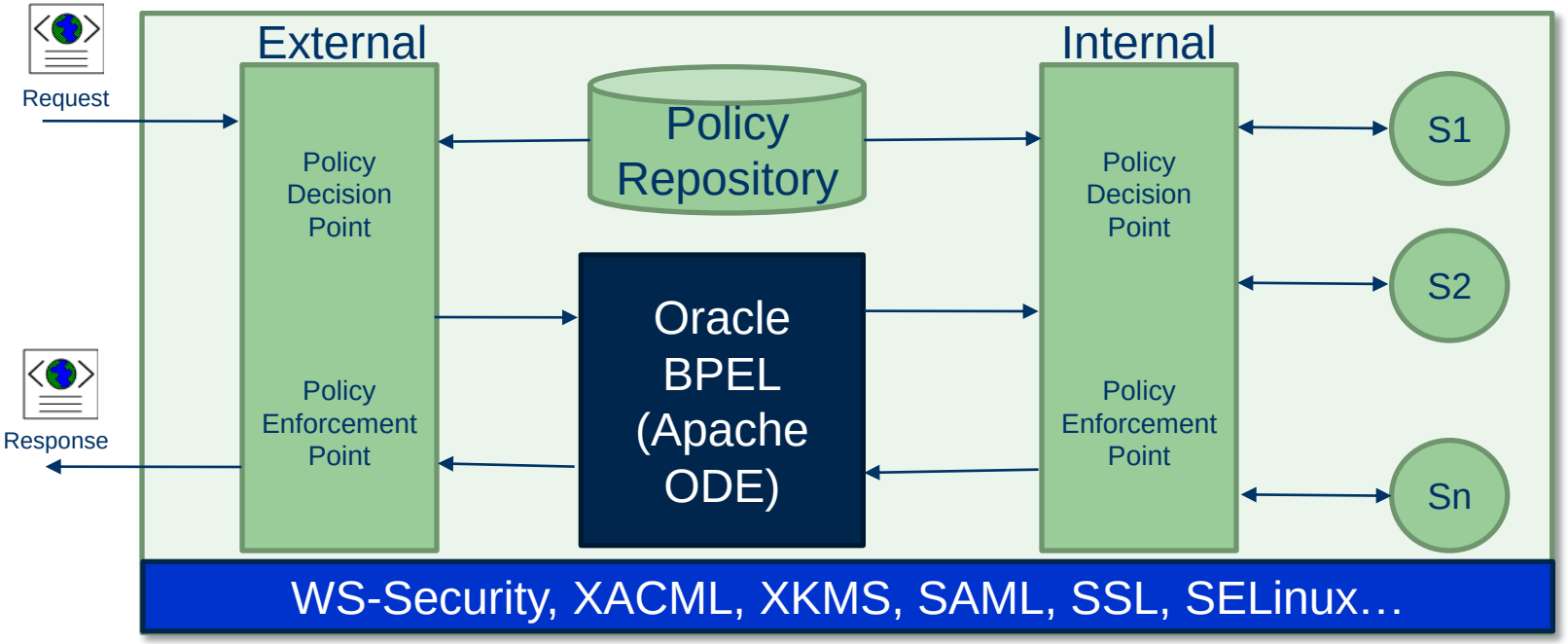
- Goal: systems design satisfying high-level requirements stated for
  - privacy, secrecy,
  - integrity,
  - non-repudiation,
  - dynamic access control,
  - rights delegation
- Last year focus
  - establishing a credible testbed for Patient Portals
  - formal modeling of **Patient Portal designs**
  - formal modeling of **access control** and **privacy policies**
  - Policy-driven control of information flows in Patient Portals

Behind the Patient Portals: Workflows and Services



Building a Credible Testbed

- Architectural Framework: SOA
  - Reliance on existing standards SOAP, WSDL, WS-Security, XACML
  - Exploiting open-source implementation of integration platforms (Active BPEL, ***Apache ODE***)



Standards do not guide integration of security technologies with applications

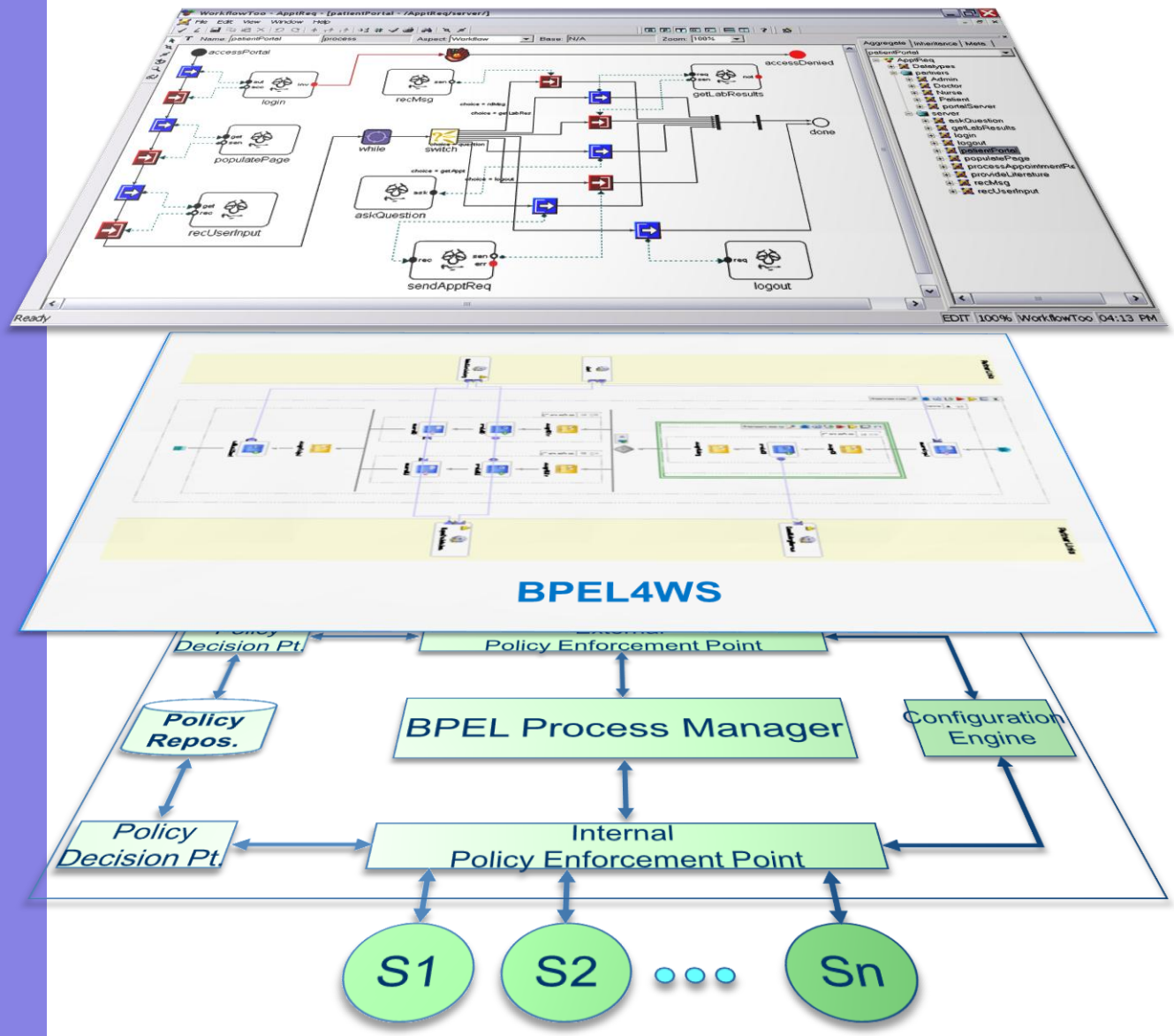
Testbed considerations

TRUST
Team for Research in Ubiquitous Secure Technology

- How to work with the Medical School?
  - They have many, complex, real, live systems that one cannot play with
  - Many legacy systems, no clear overarching architecture
  - The selected platform has to be viable for a wide range of future systems
- SOA can fulfill this role
  - Proven standards-based technology successfully applied in many different domains
  - It enables experimentation with different techniques to deal with security/privacy issues
  - By building on the existing massive infrastructure we can focus on interesting research issues and not on the technical details of the really complex machinery behind it
- Focus: how to build applications
  - How to specify security/privacy requirements and
  - How to tie them to the underlying standard technologies

  Because the standards do not provide guidance on how to integrate security/privacy technologies with applications
- Value added:
  - Mature, proven, model-based tool environment
  - Automatic generation of many required artifacts
  - New components for policy specification/evaluation

Abstraction Layers

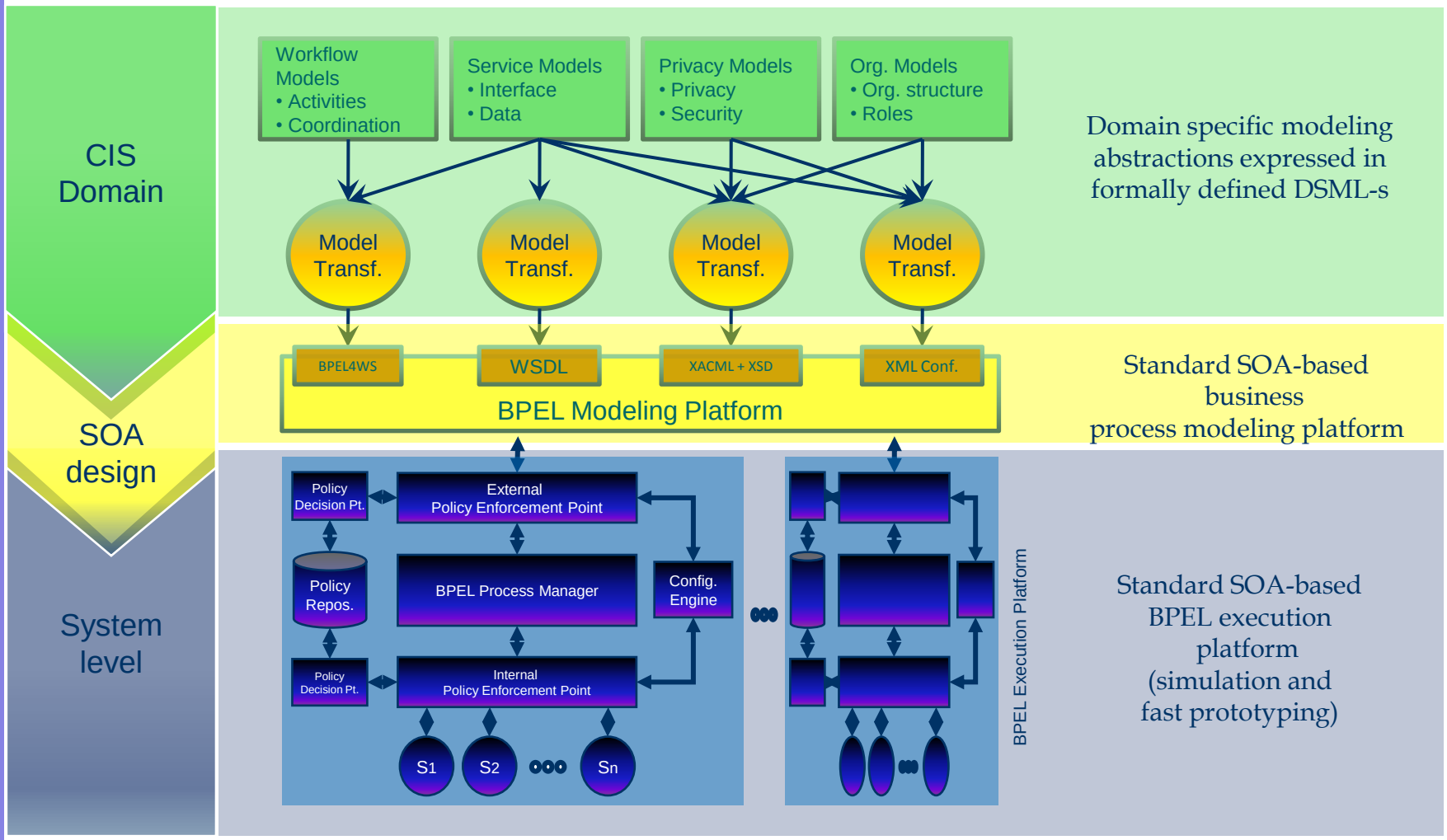


Domain specific modeling abstractions expressed in formally defined DSML-s. TRUST research focus

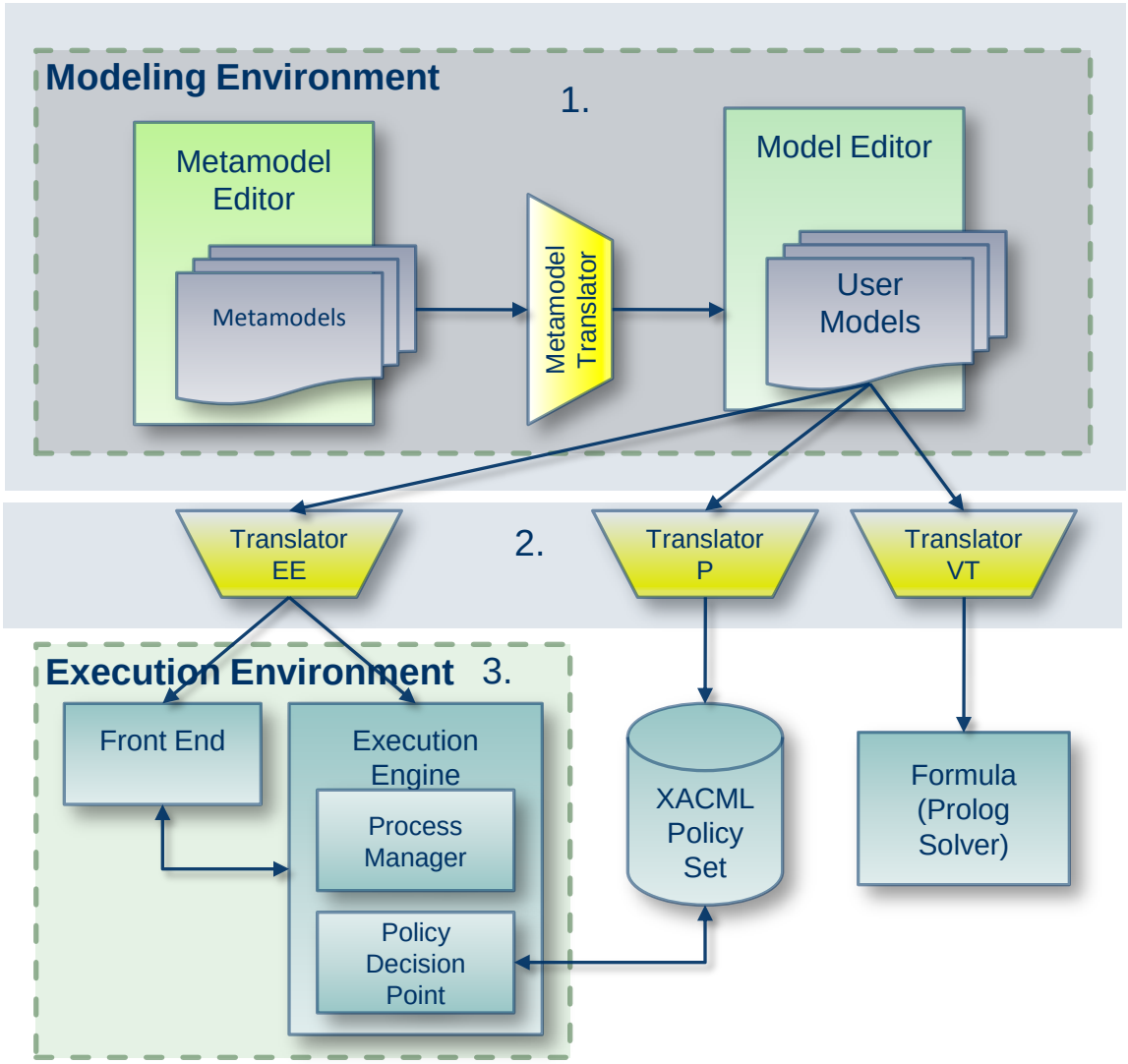
SOA-based, standard, business process modeling platform

SOA-based, standard, execution platform (simulation/fast proto.tng)

Architecture



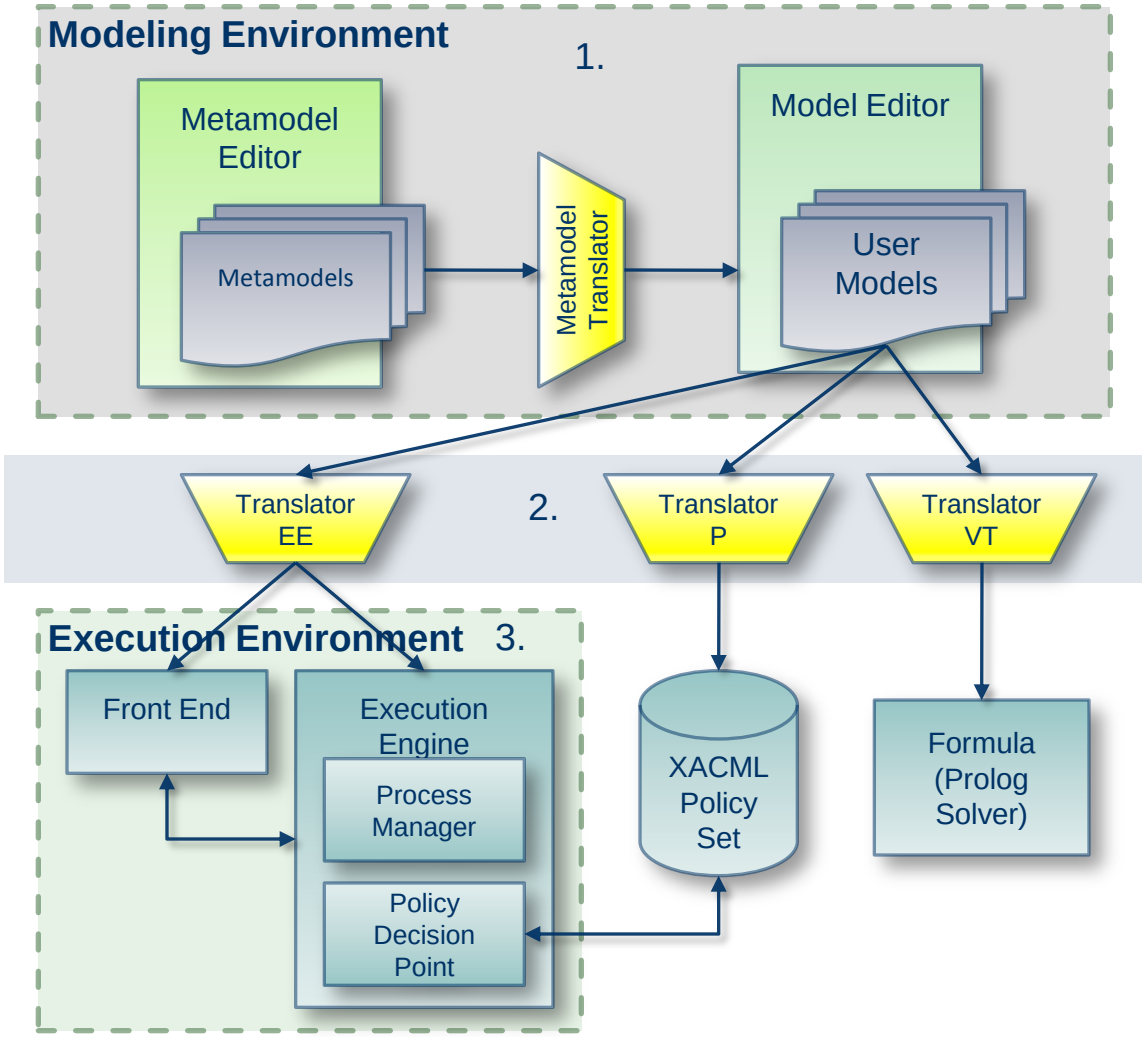
MICIS Architecture



1. Modeling environment
   - Metamodels define the domain specific modeling language and define the abstract syntax of domain models
   - User models represent a specific CIS instance through a set of modeling abstractions

- Using
  - Generic Modeling Environment (GME)

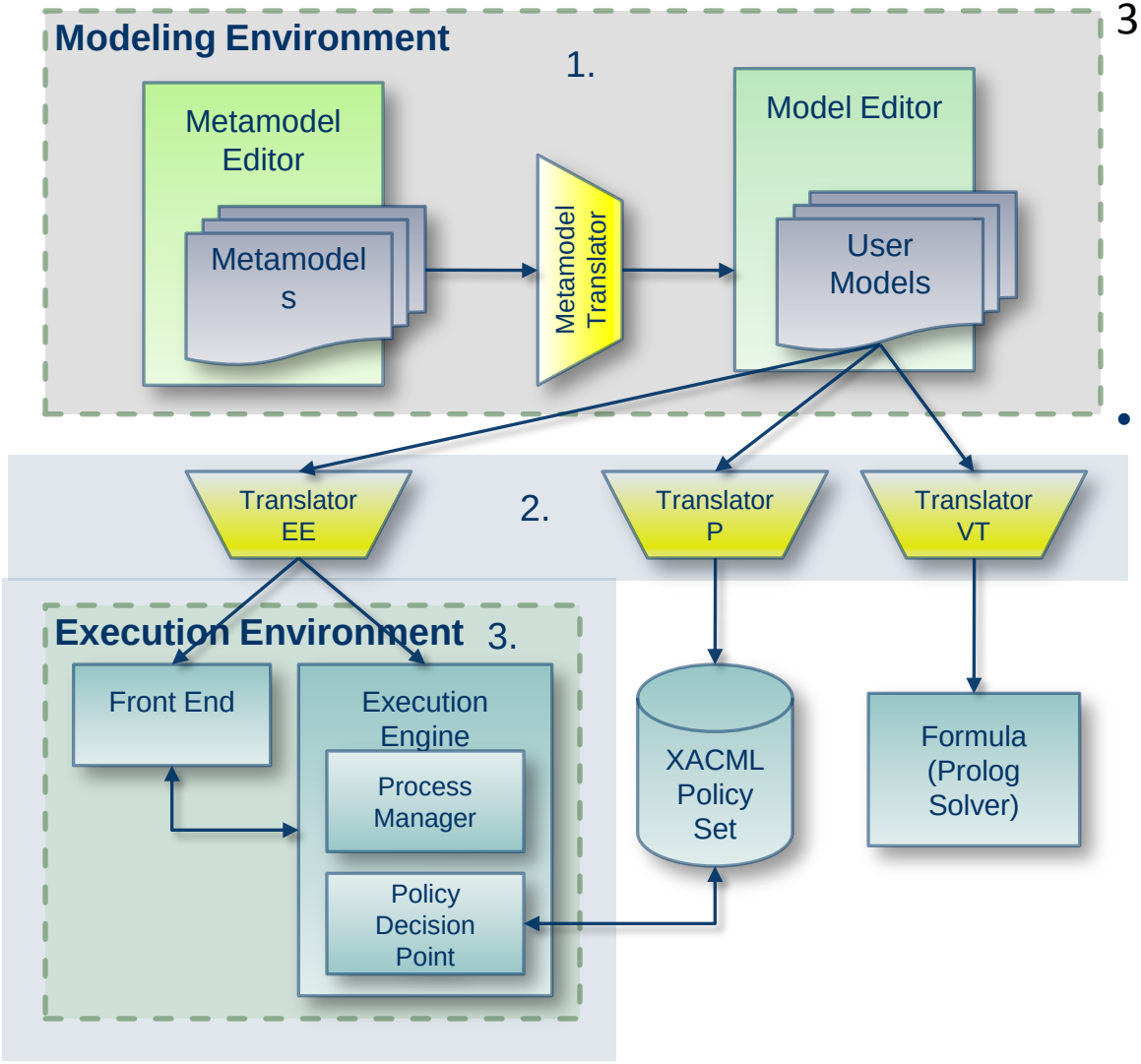
MICIS Architecture



2. Translators
   - Transform user models into BPEL deployment code
   - Create XACML policy decision points

- Using
  - GREAT
  - Builder Object Network (BON) interface

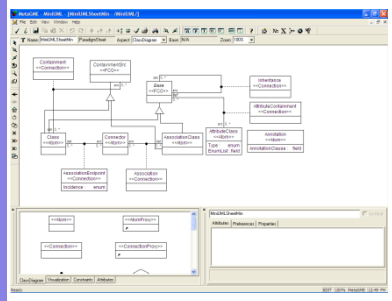
MICIS Architecture

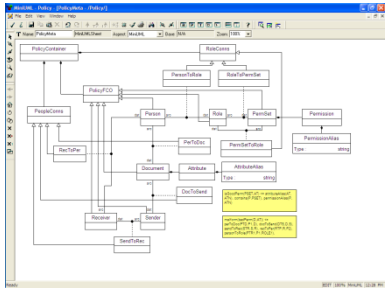


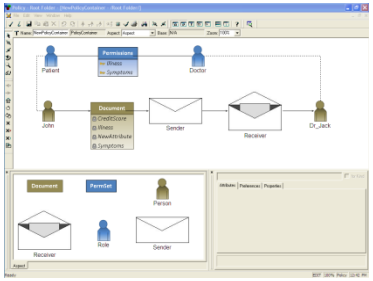
3. Execution Environment
   - BPEL execution engine
   - Policy execution engine
   - Web server for user interaction

- Using
  - OracleBPEL
  - AciveBPEL
  - SunXACML

Policy Verification/Enforcement w/ Prolog







MiniUML-in MetaGME

T_0

DSML in MiniUML

T_0

Design Model in DSML

T_n

T_m

T_1

T_1

T_1

Prolog

Prolog

Prolog

Prolog representation of the MiniUML domain

Prolog representation of the DSML domain

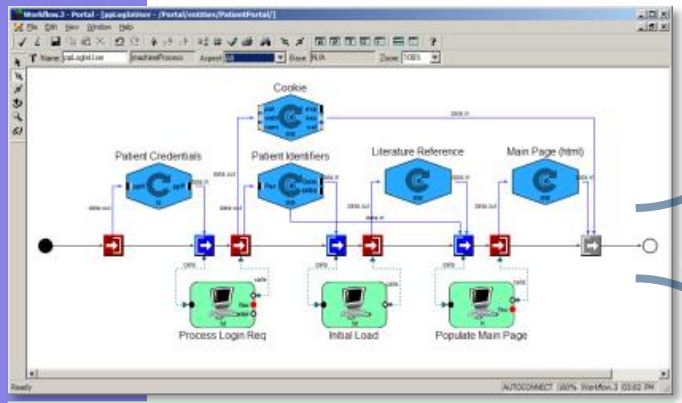
Prolog representation of the Design model

Generate domain

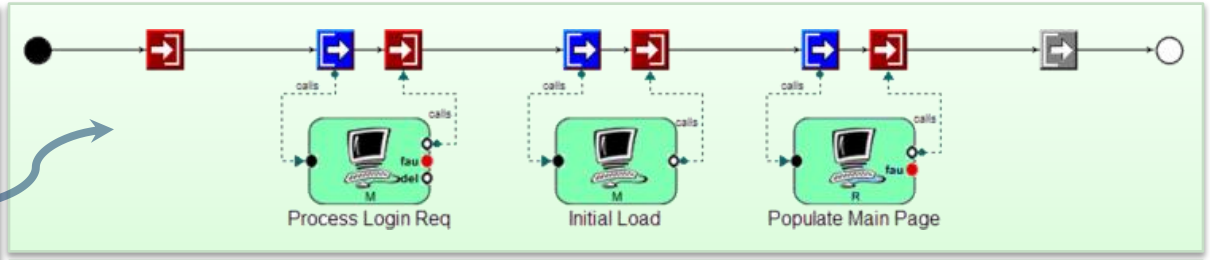
Generate domain

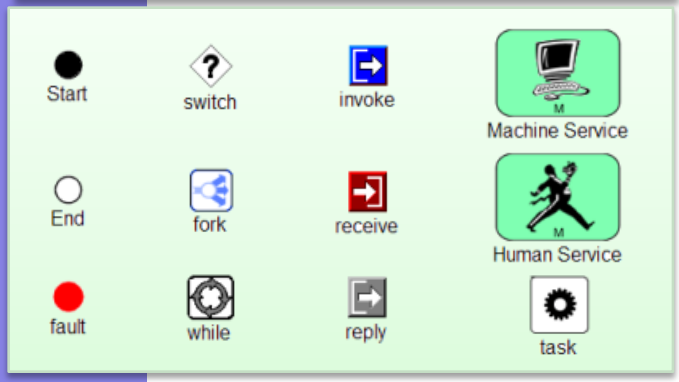
MICIS Modeling Abstractions

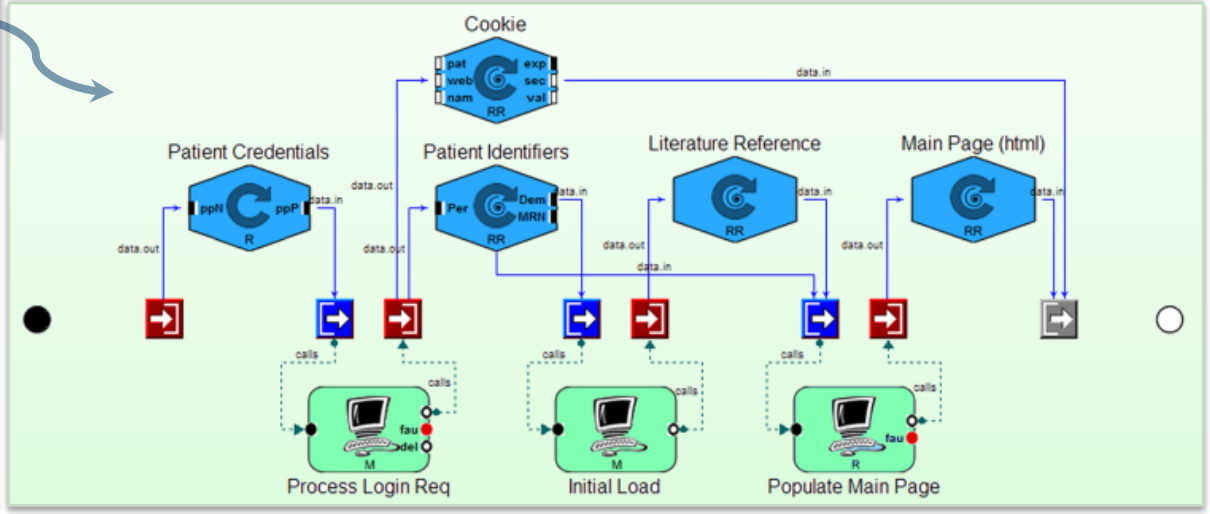
Design Environment (GME) – Combined View



Control Flow View





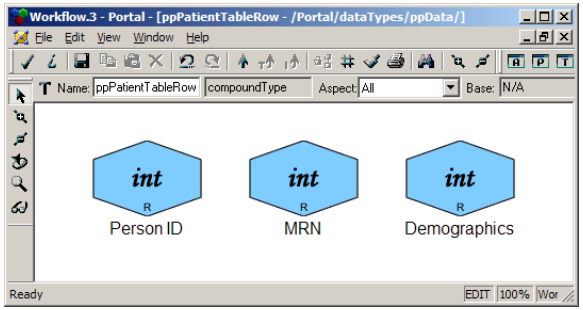


Component View

Data Flow View

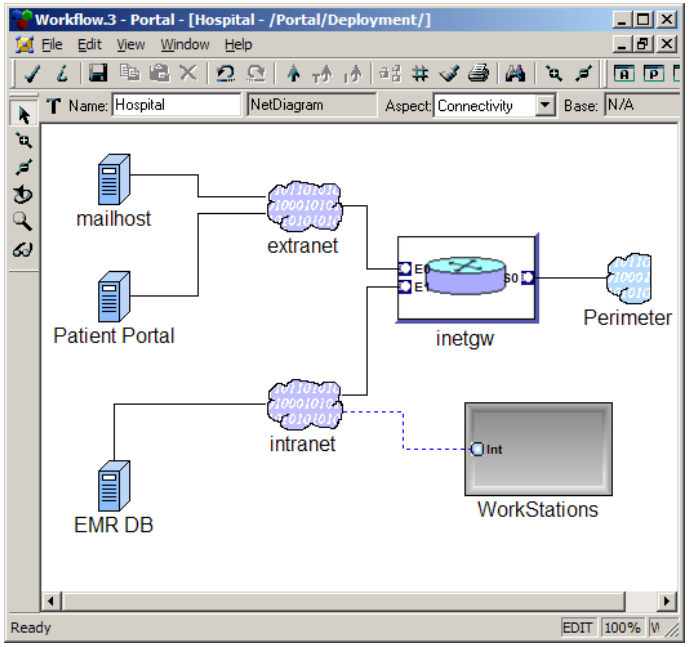
- Service models capture business logic
  - Control flow
  - Data flow

MICIS Modeling Abstractions



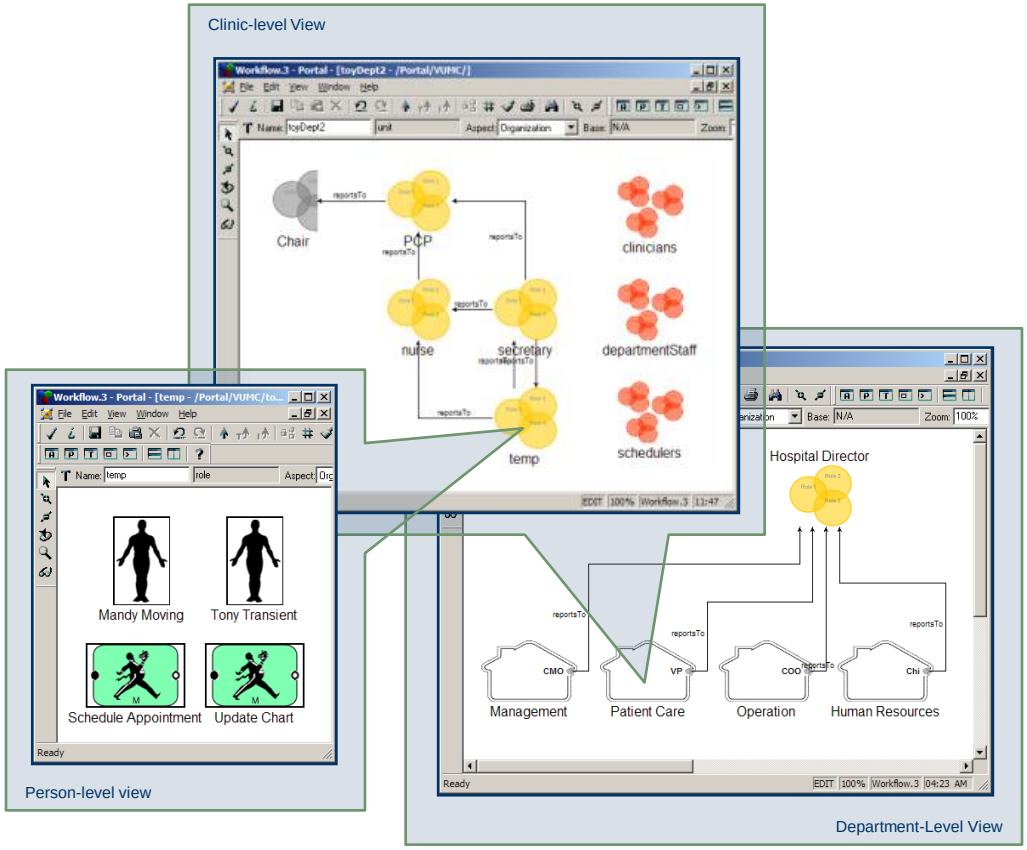
- Data models
  - Specify the information in the CIS
  - Represent patient information and system state variables
  - Simple and compound data types in hierarchy

MICIS Modeling Abstractions



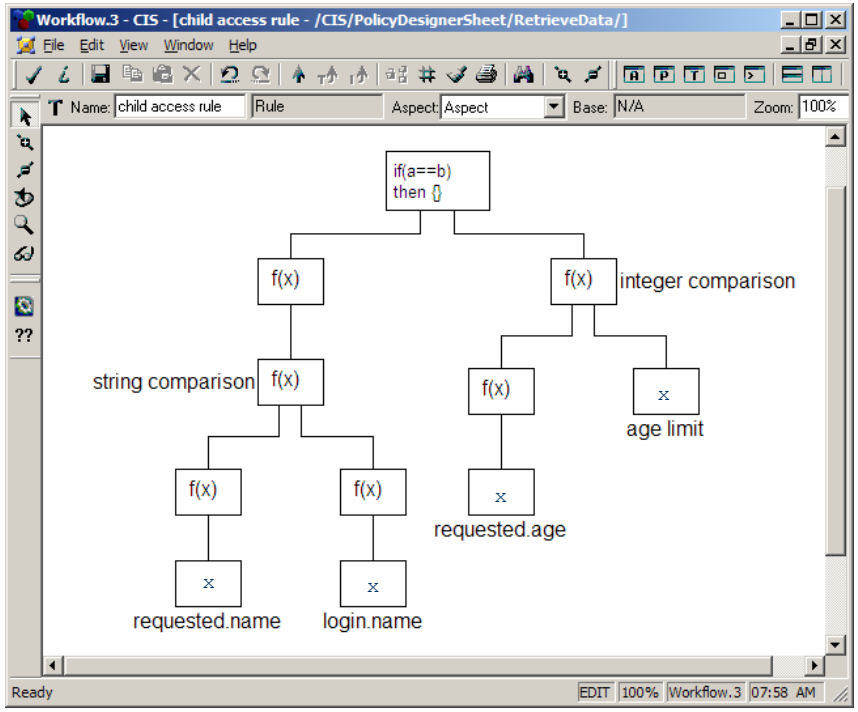
- Deployment models
  - Servers and workstations
  - Service deployment
  - Secure sessions
  - Access control

MICIS Modeling Abstractions



- Organizational models
  - Specify human coordination within CIS
  - Roles, groups of roles and people within clinics
  - Interaction between roles

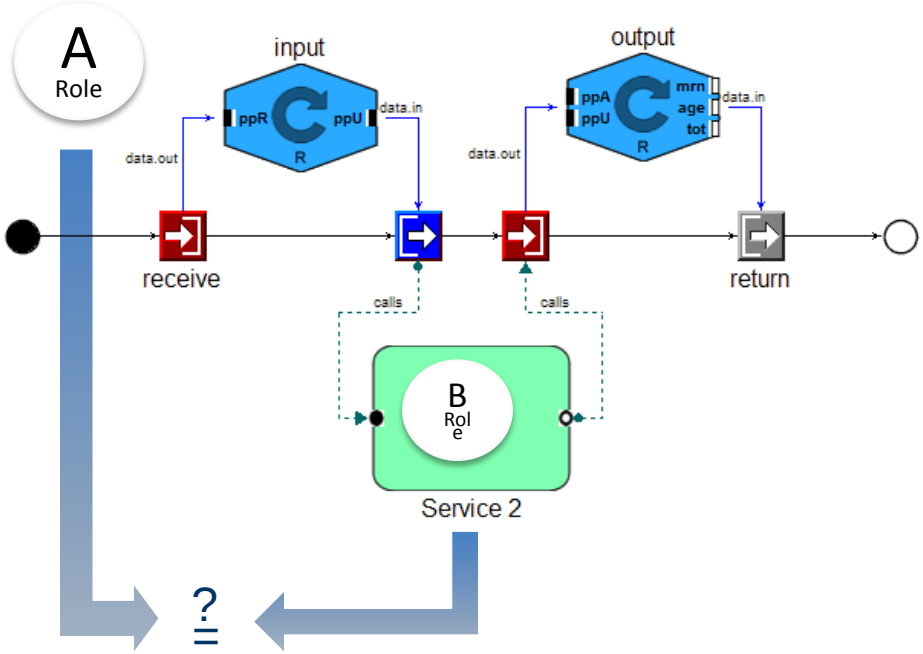
MICIS Modeling Abstractions



- Policy models
  - Static policies that can be evaluated based on system specifications
  - Dynamic policies that must be evaluated at run-time

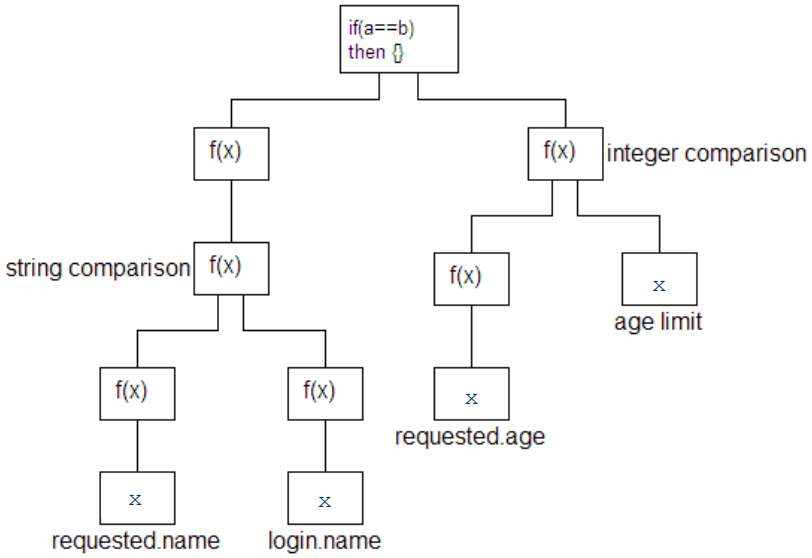
Policy Models

- Static policy models



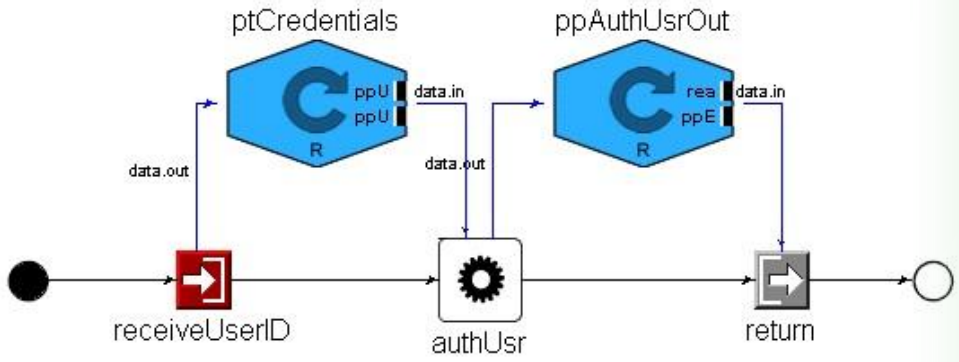
  - Ex: A service's workflow and an invoked service have to have matching roles

- Dynamic policy models
  - Ex: Check if the patient whose record is being retrieved is under a certain age




```
[ ( requested.name != login.name ) && ( requested.age < 18 ) ]
```

MICIS Example



Above:

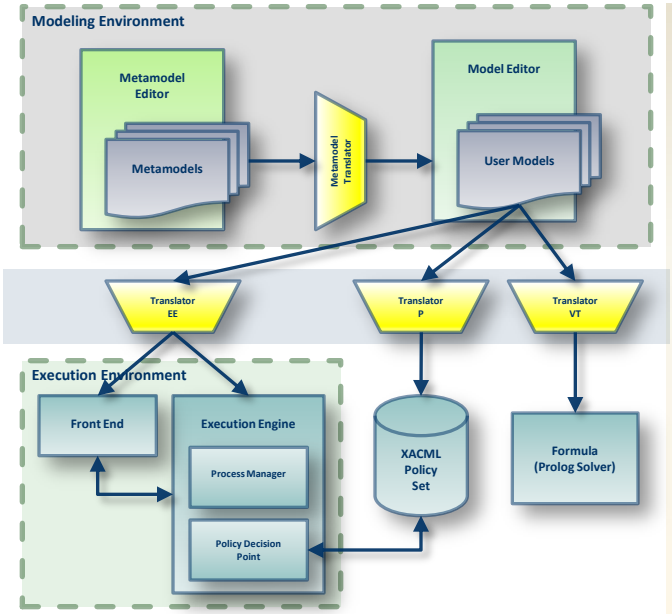
- a simple service that checks the user's credentials and authorizes access to other services

Symbol Guide

MICIS Demo

Conclusions

- MICIS forms a testbed for EMR research
  - Helping to solve privacy and security challenges of EMR *systems* applications
  - It can be used for the integration, testing and evaluation of new technologies



Future work

- Prolog-based policy management
  - Choice of policy languages: XACML, Prolog, OCL, ???
  - How to structure policies
  - Static vs. dynamic
- Analysis/verification tools