



Experimental Platform for Model-Integrated Clinical Information Systems

Janos Mathe[‡], Jan Werner[‡], Yonghwan Lee[‡],
Akos Ledeczki[‡], **Bradley Malin^{‡#}**, Janos Sztipanovits[‡]

[‡]Department of Electrical Engineering and Computer Science
[#]Department of Biomedical Informatics
Vanderbilt University

Carnegie Mellon

Cornell University

MILLS
COLLEGE

San José State
UNIVERSITY



STANFORD
UNIVERSITY

Berkeley
UNIVERSITY OF CALIFORNIA



- Electronic Medical Records (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 core TRUST research areas, including:
 - Model-based design for security and privacy
 - Formal modeling, verification, enforcement of privacy & security policies
 - Data mining & representation of real clinical workflows
 - Security & privacy technologies for sensor networks
 - Public policy to technology interactions

1. Experimental platform for Model-Integrated Clinical Information Systems (MICIS)
 - Provide a common integration testbed for security and privacy aware Clinical Information Systems (CIS).

2. Component integration platform
 - Based on a standard Service-Oriented Architecture framework (SOA)
 - Extended Prolog-based Policy Evaluation Point & Policy Enforcement Point components (MICIS-PROPER)
 - *Reusable*
 - *Platform-Independent*
 - Integrated with the Apache Orchestration Director Engine (ODE)

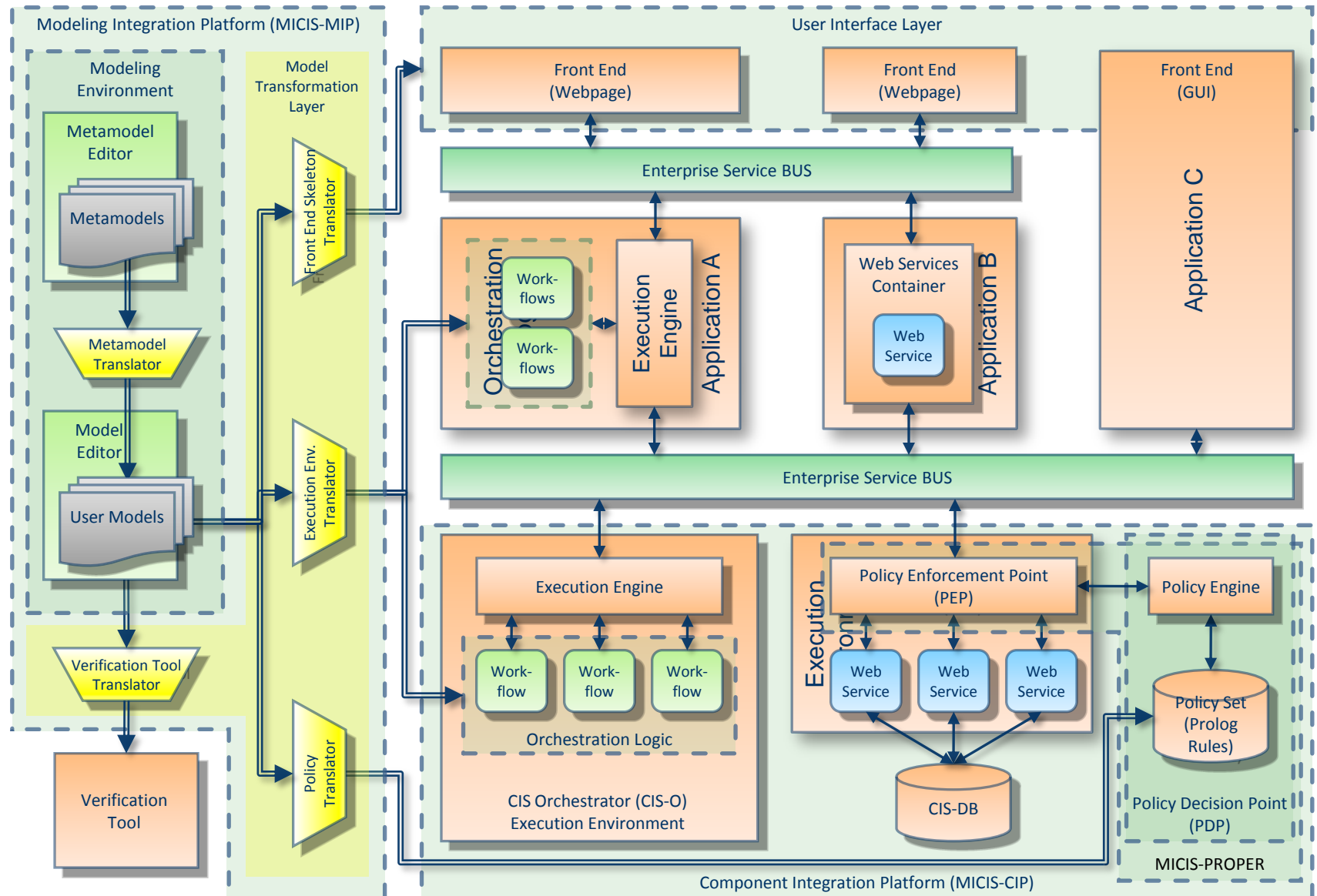
3. Model integration platform

- Built on Vanderbilt's metaprogrammable Model-Integrated Computing (MIC) tool suite
- System models capture environment
 - *Workflows*
 - *Services*
 - *Deployment*
 - *Messages*
 - *Message Attributes*
 - *Organizations*
 - *Roles*
 - *Access control policies*
 - *Security policies*
- Privacy modeling language based on Stanford's work on contextual integrity
 - Enables formal representation of permitted communications
 - Considers past, as well as future, communication instances

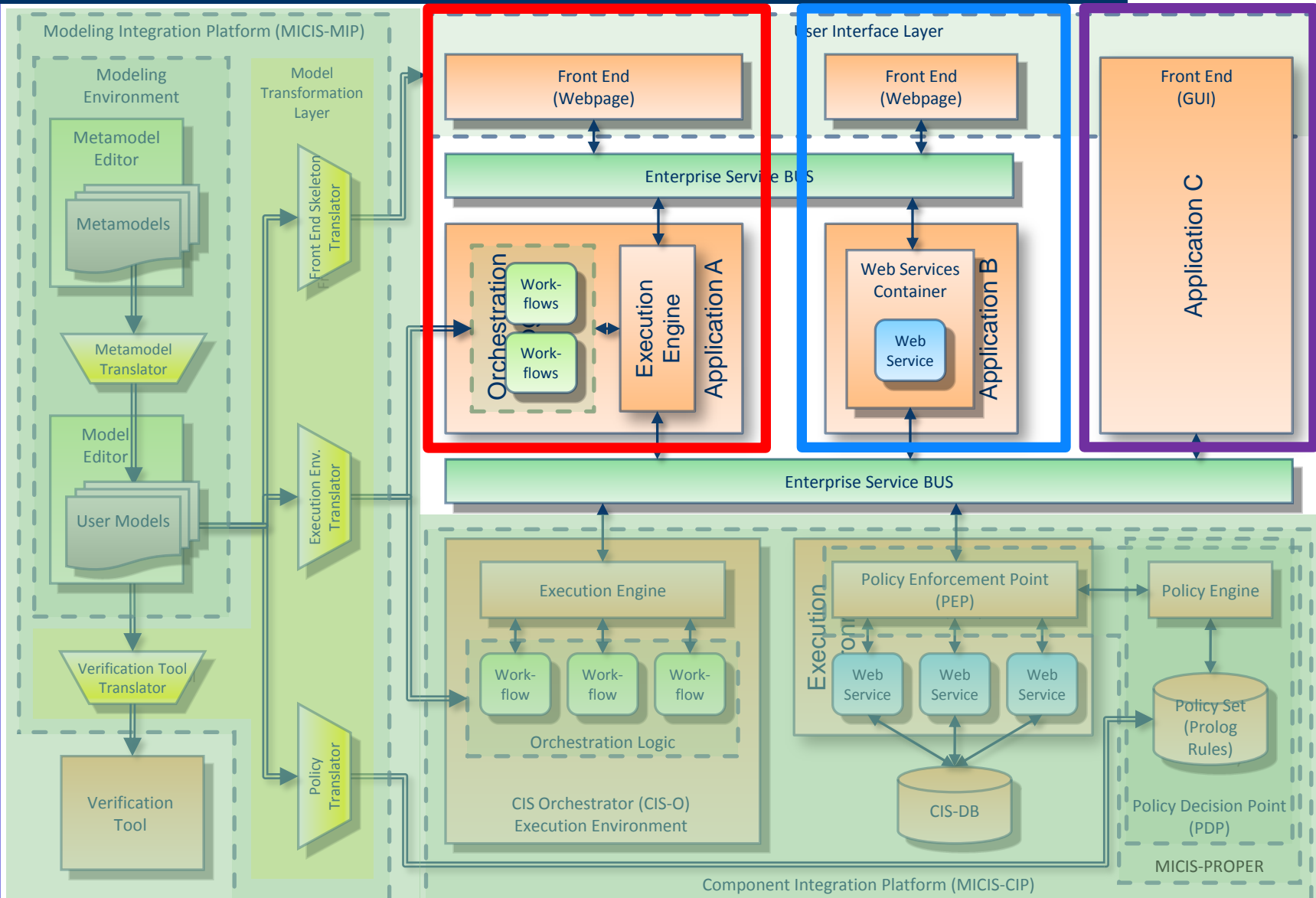
3. Model integration platform

- Experimental platform has several components:
 - Set of domain-specific modeling languages
 - Captures relevant architectural components
 - Captures policy modeling aspects of selected CIS applications
 - Model transformations
 - Map domain-specific models on the MICIS component integration platform
 - Example application models
 - Running experiments for analytic analysis

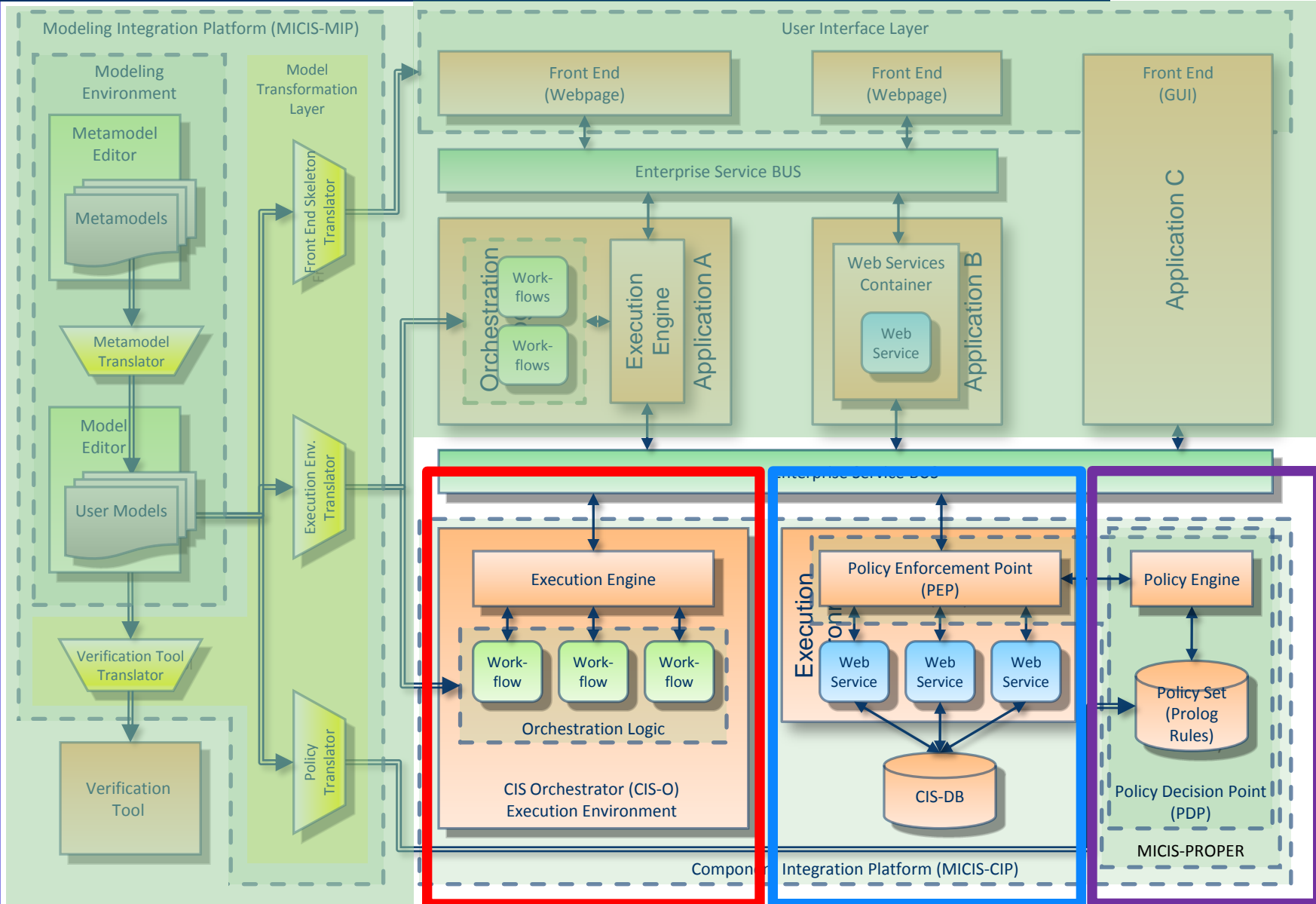
Architecture (Big Picture)



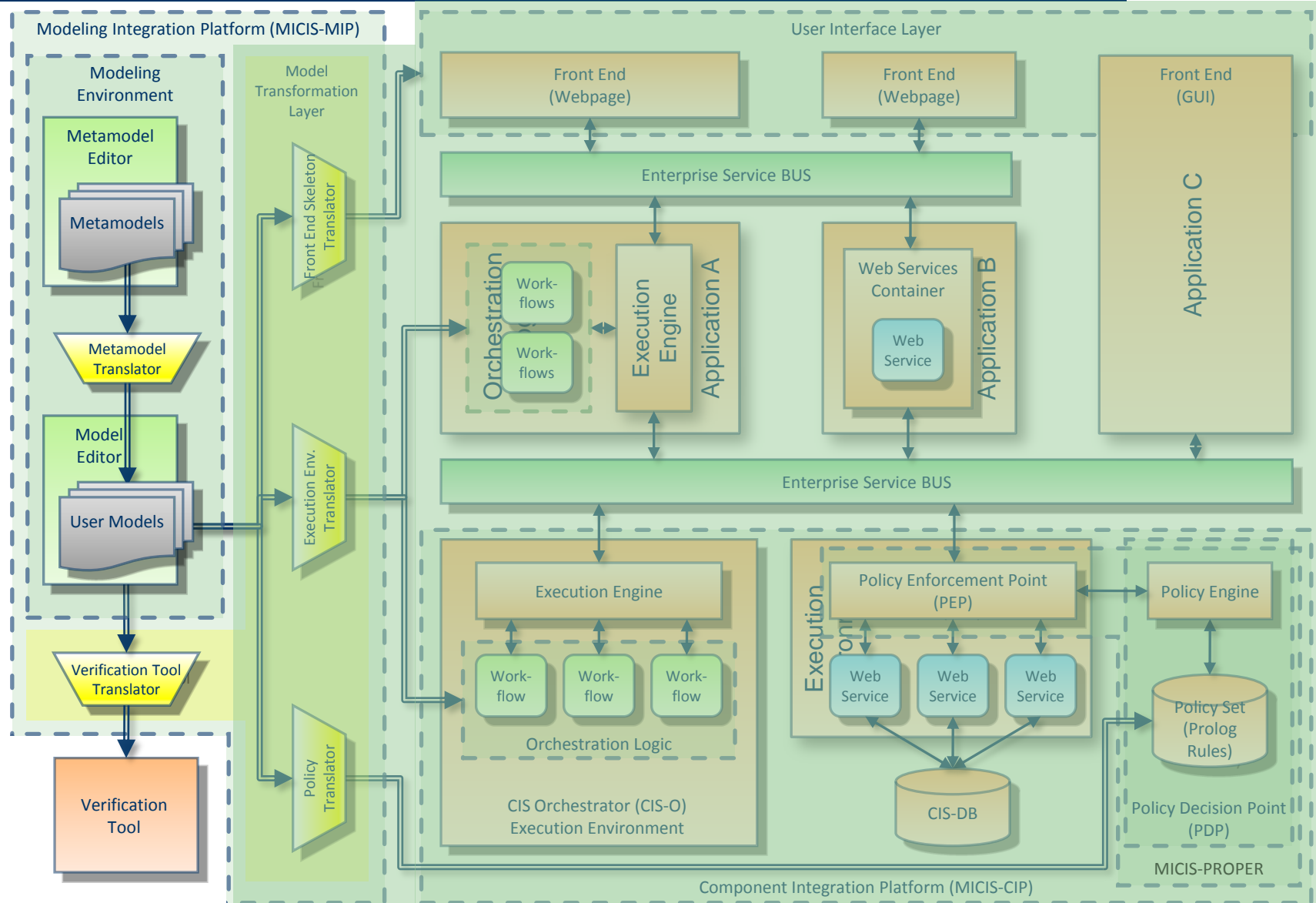
Architecture: Applications



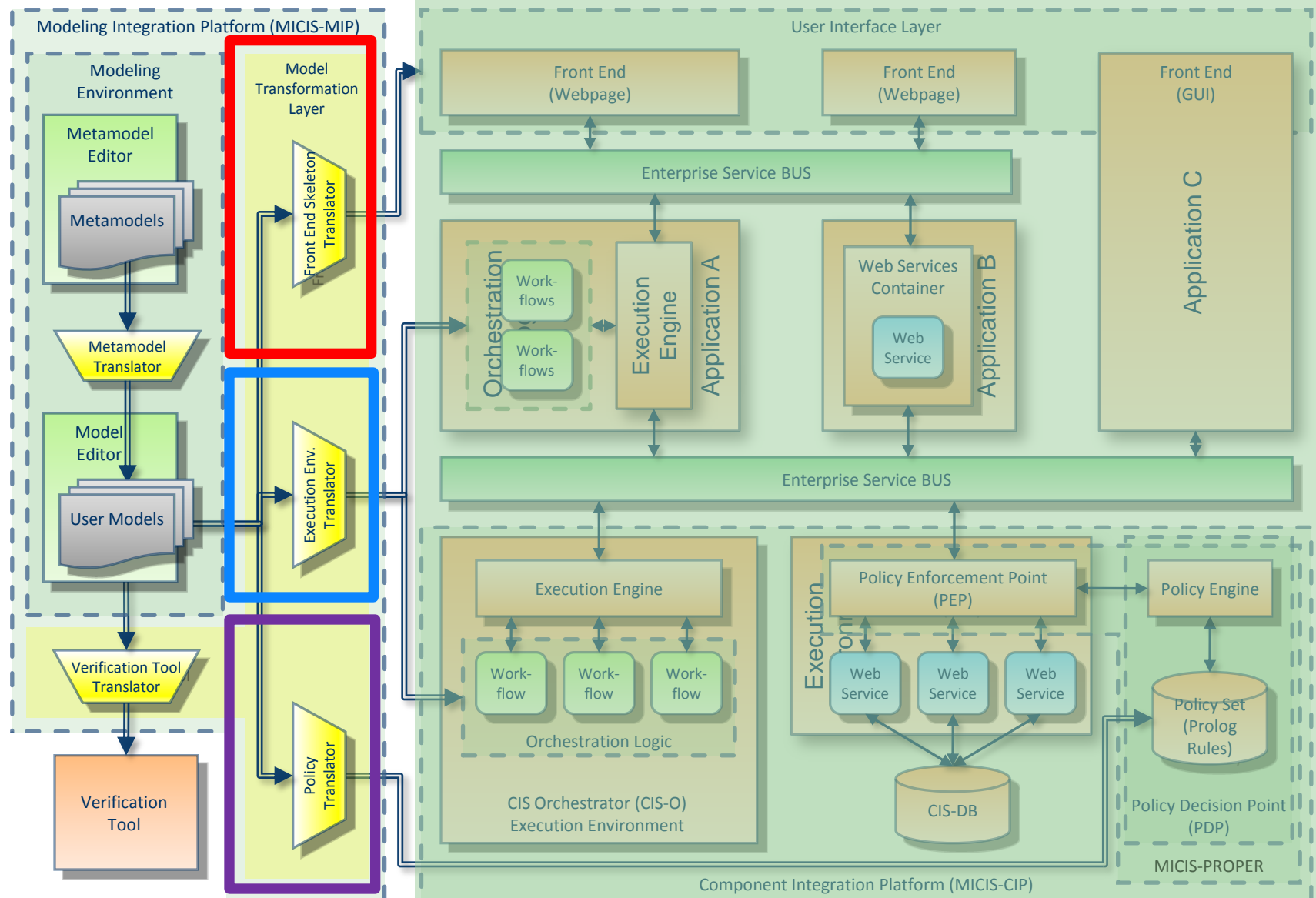
Architecture: Execution / Control



Architecture: Modeling

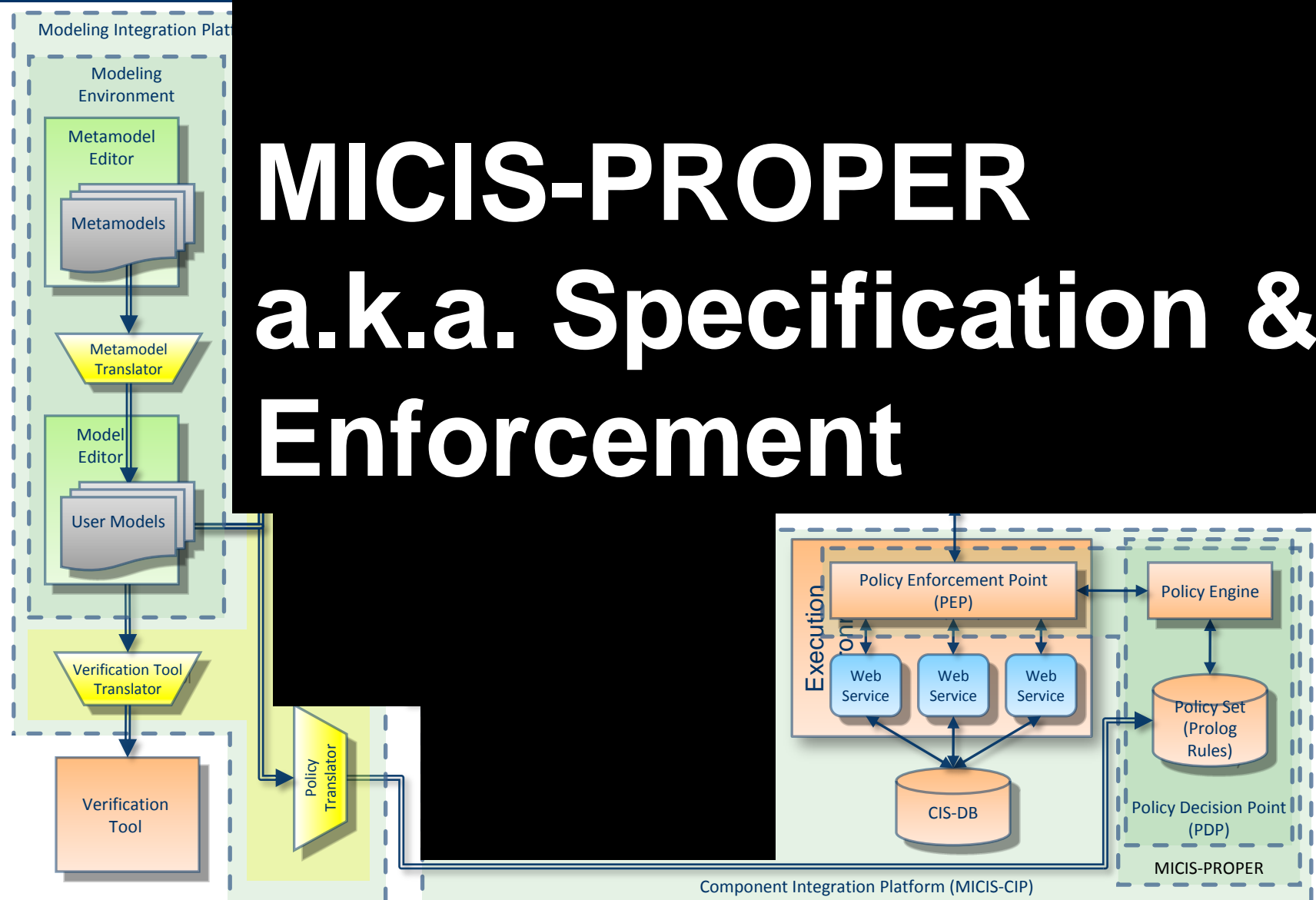


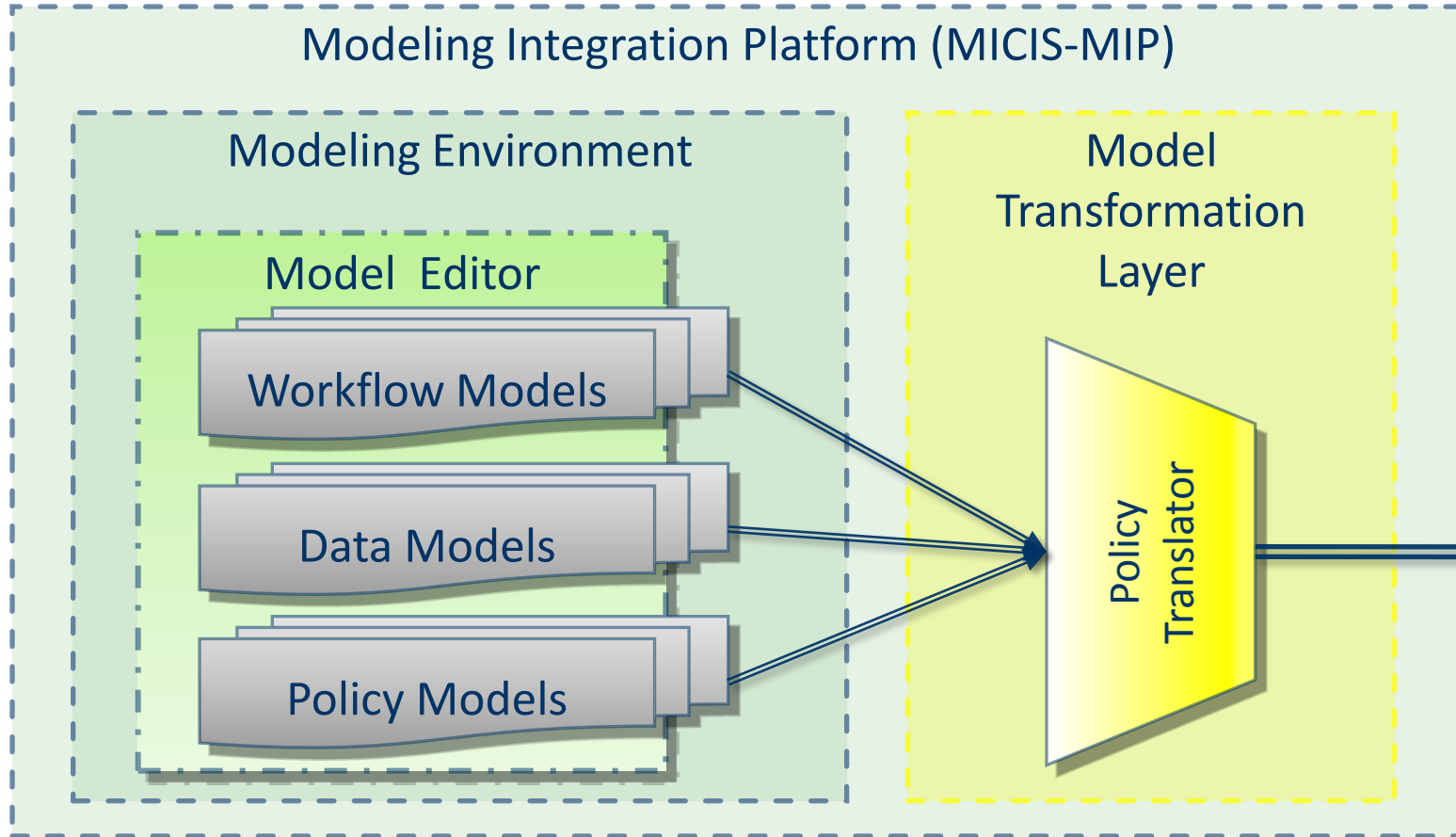
Architecture: Model Transforms



Architecture: Model Transforms

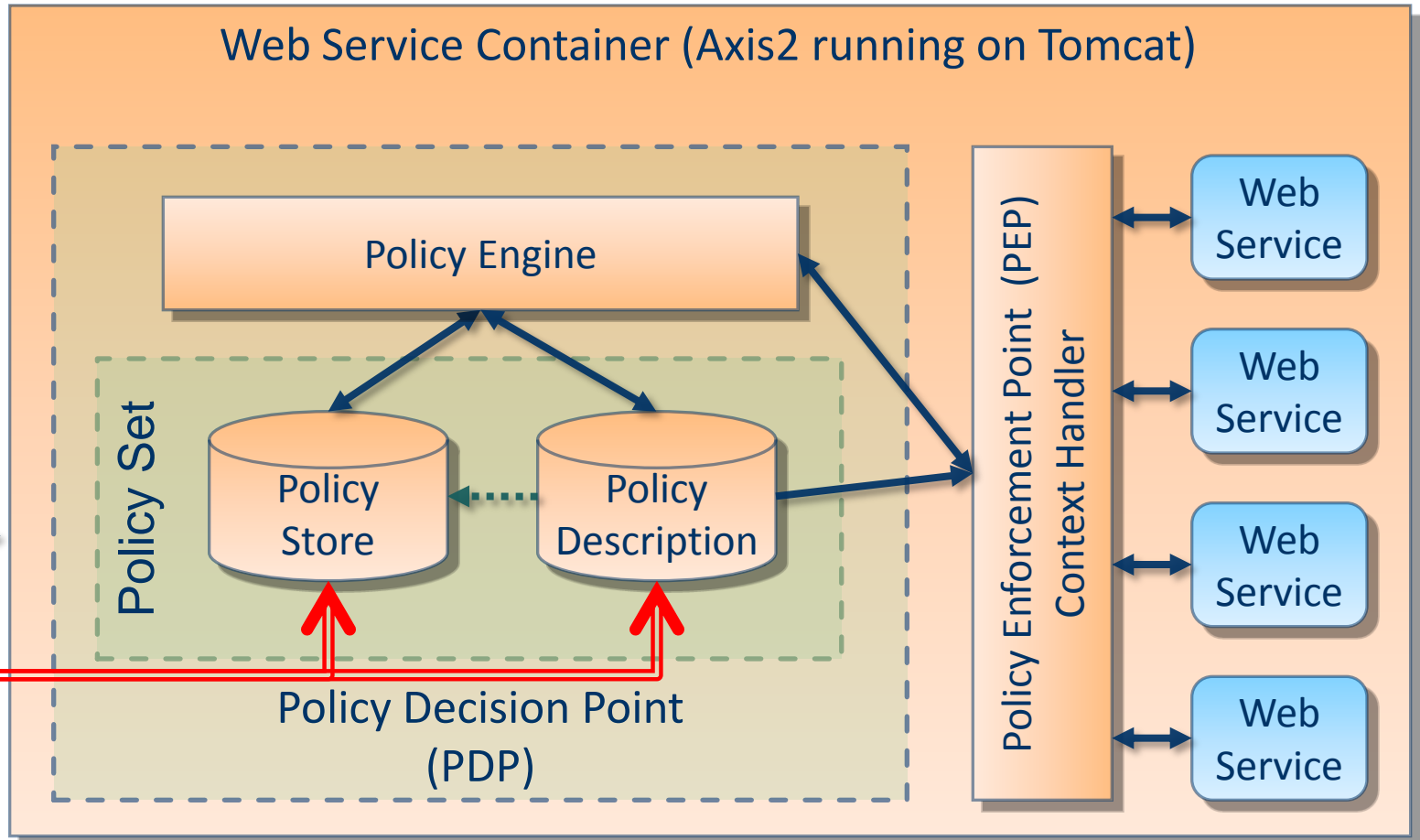
MICIS-PROPER a.k.a. Specification & Enforcement





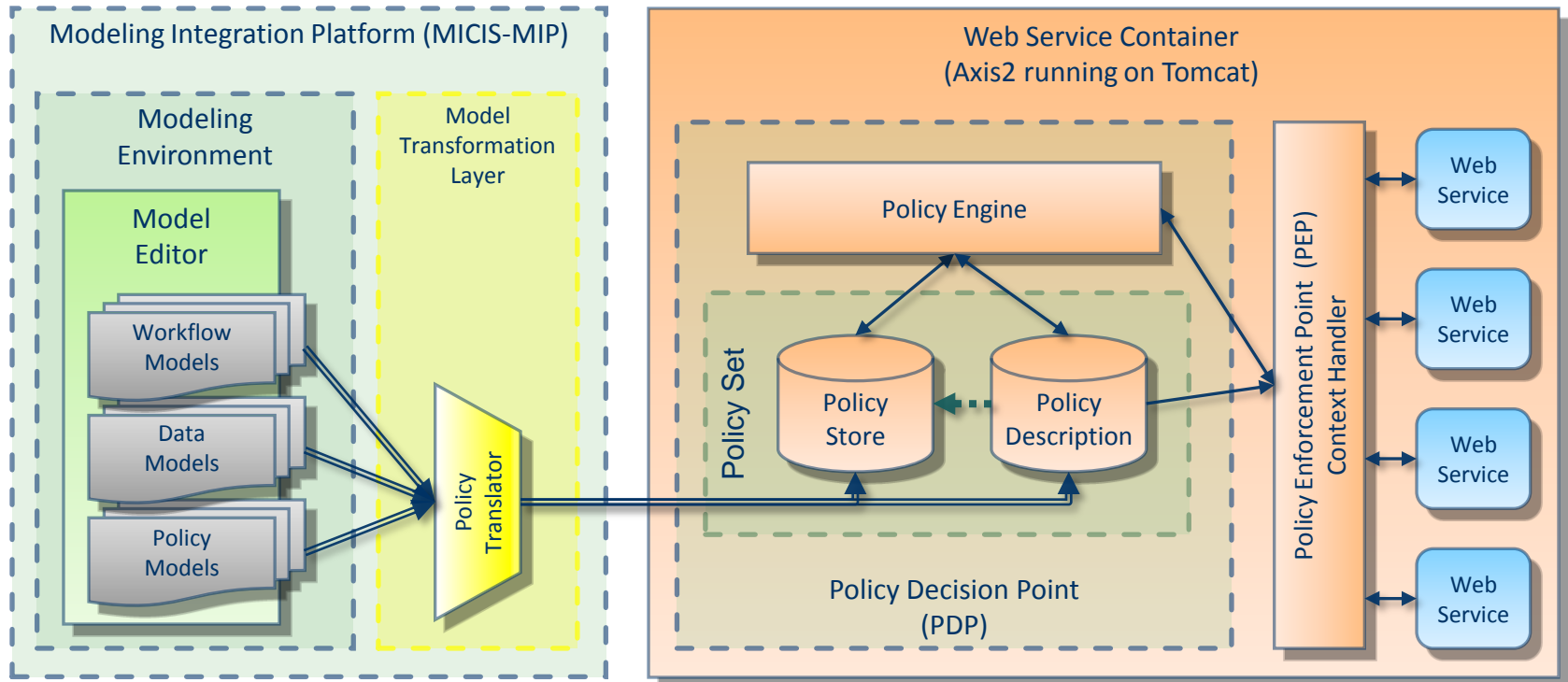
Prolog-based Policy Evaluation Point and Policy Enforcement Point (MICIS-PROPER)

MICIS-PROPER architecture

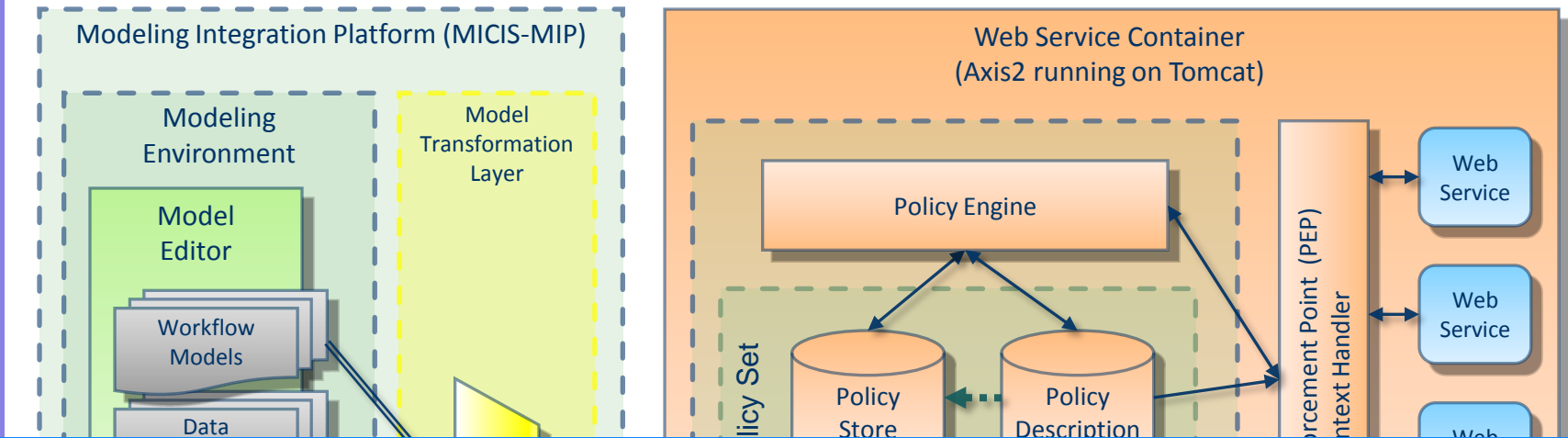


Prolog-based Policy Evaluation Point and Policy Enforcement Point (MICIS-PROPER)

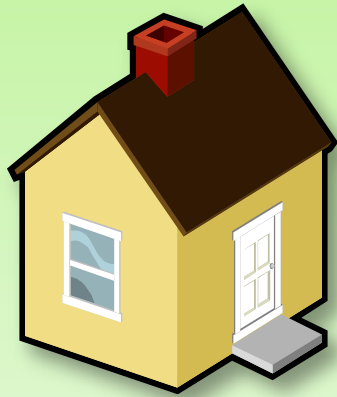
MICIS-PROPER architecture



MICIS-PROPER architecture



- Integrated with Apache Orchestration Director Engine (ODE)
- Enabler
 - construct rigorous specification via privacy & security languages
 - experimental analysis of specification in complex system
 - description of security and privacy constraints with temporal aspects
 - rich user-defined contextual dependence



Outpatient

Outpatient monitoring
system

Wearable sensors, video
capture, wireless networking

TRUST Project:
Berkeley
Cornell
Vanderbilt

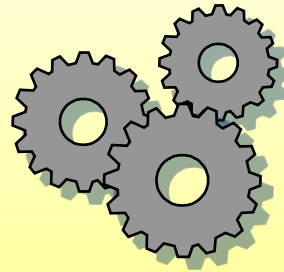
Tying it Together: An Example Scenario



Outpatient

Outpatient
monitoring system

1. send AlertMessage



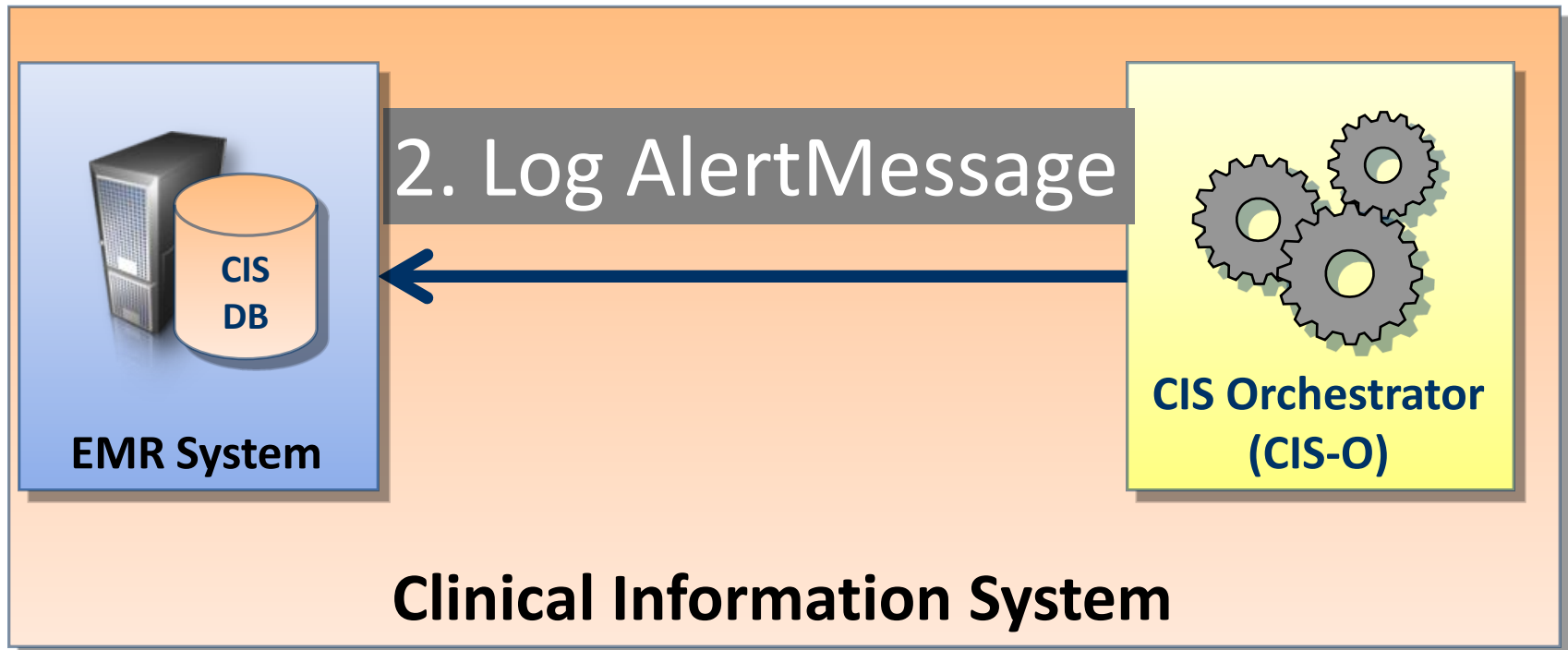
**CIS Orchestrator
(CIS-O)**

Clinical Information System

Tying it Together: An Example Scenario



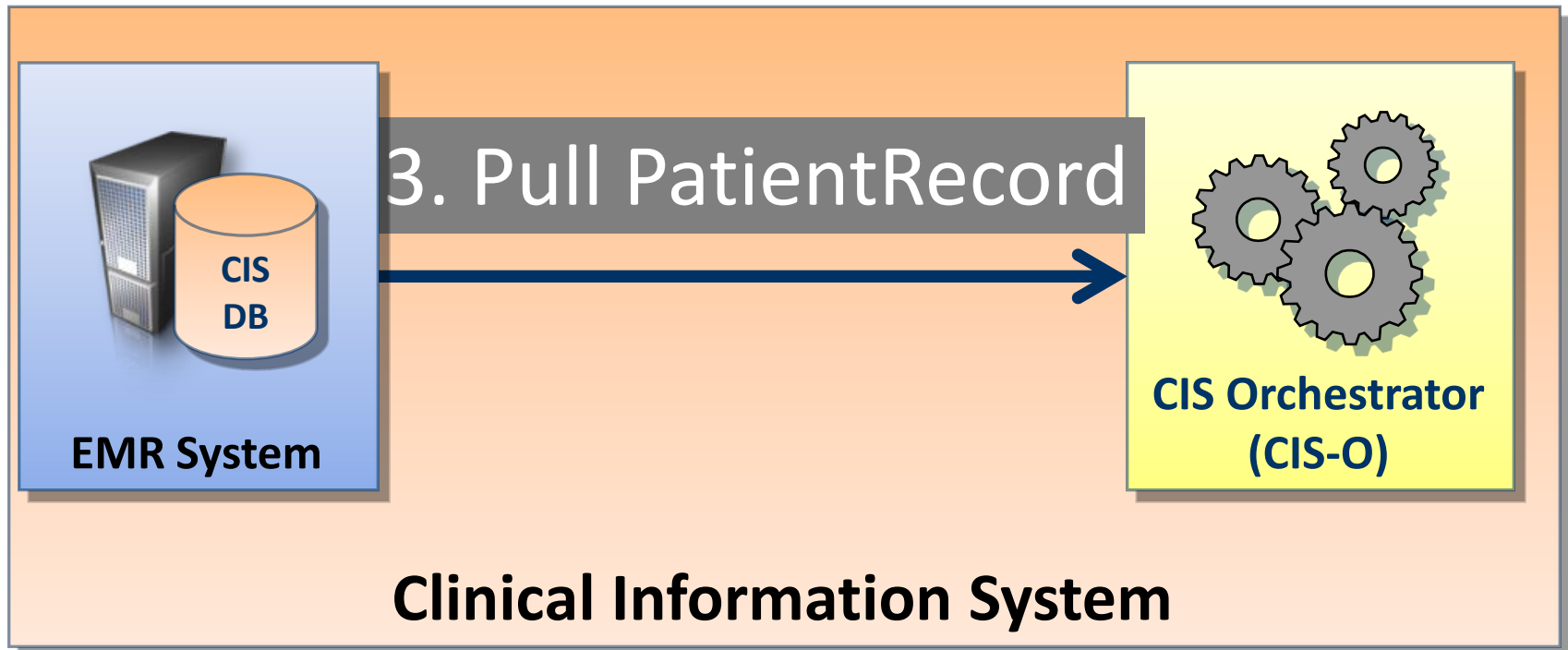
Outpatient monitoring system



Tying it Together: An Example Scenario



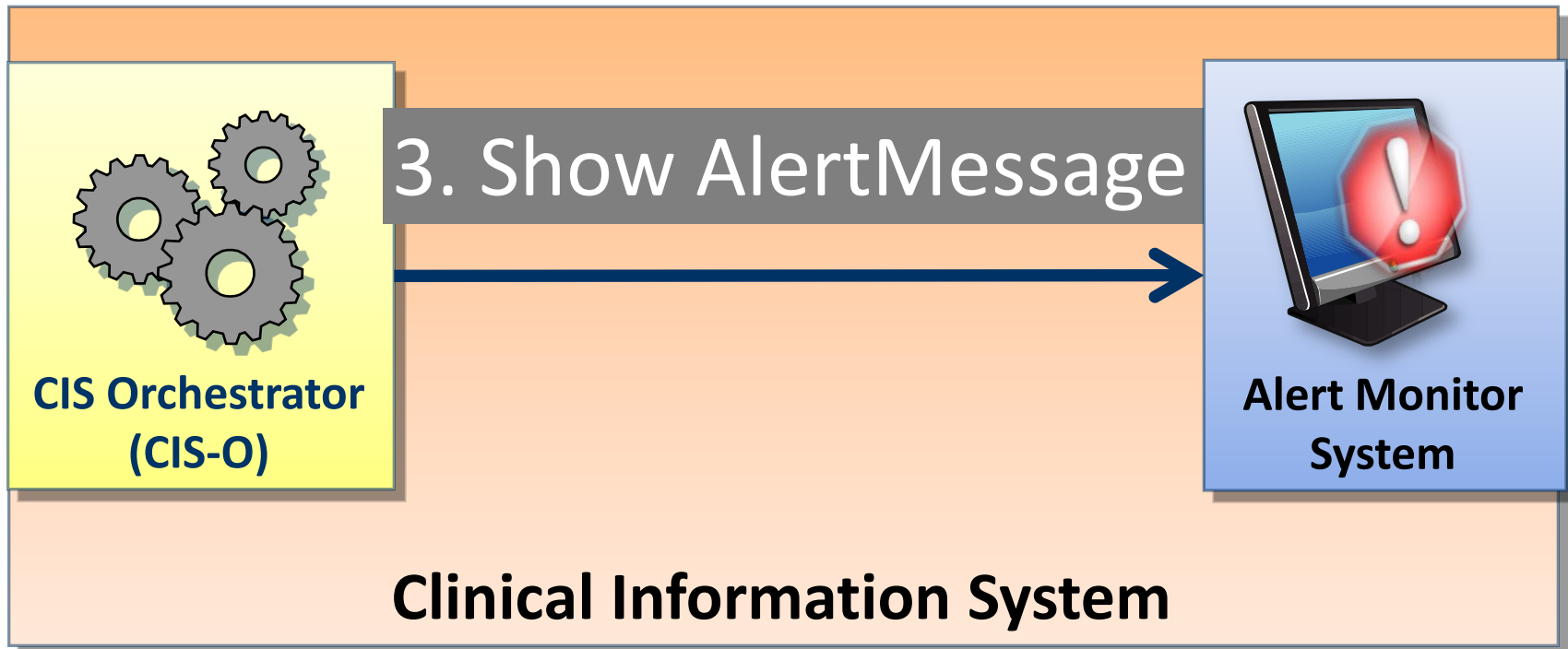
Outpatient monitoring system



Tying it Together: An Example Scenario



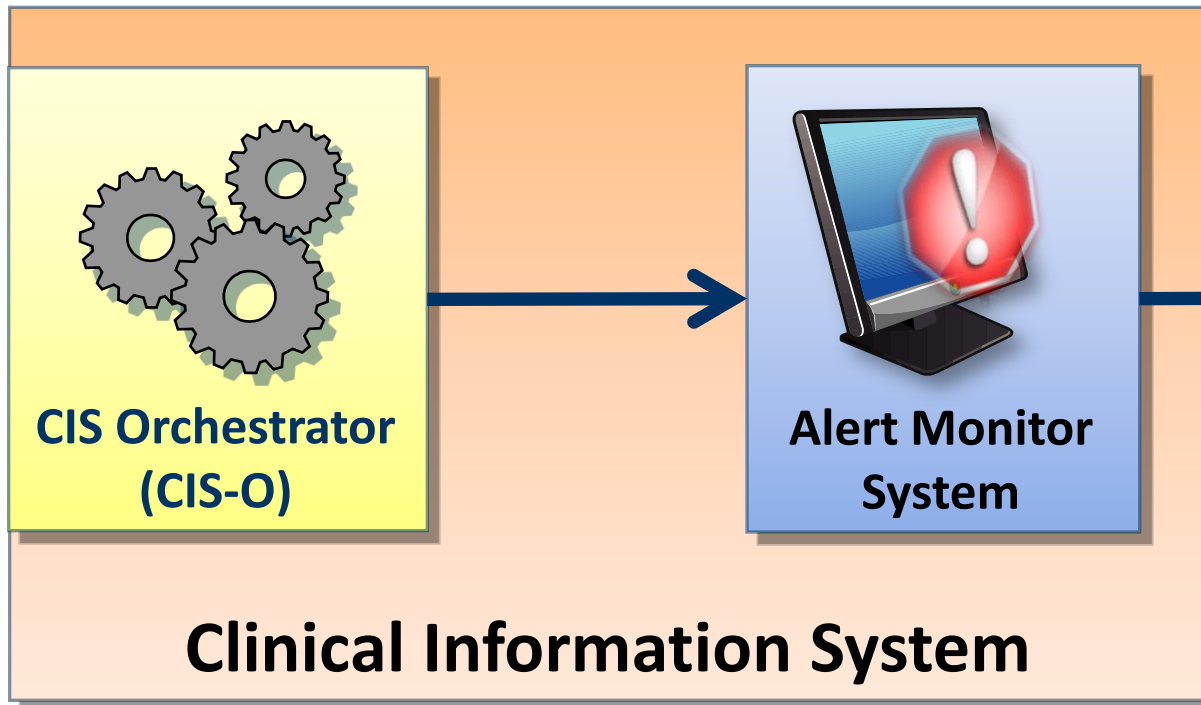
Outpatient monitoring system



Tying it Together: An Example Scenario



Outpatient monitoring system



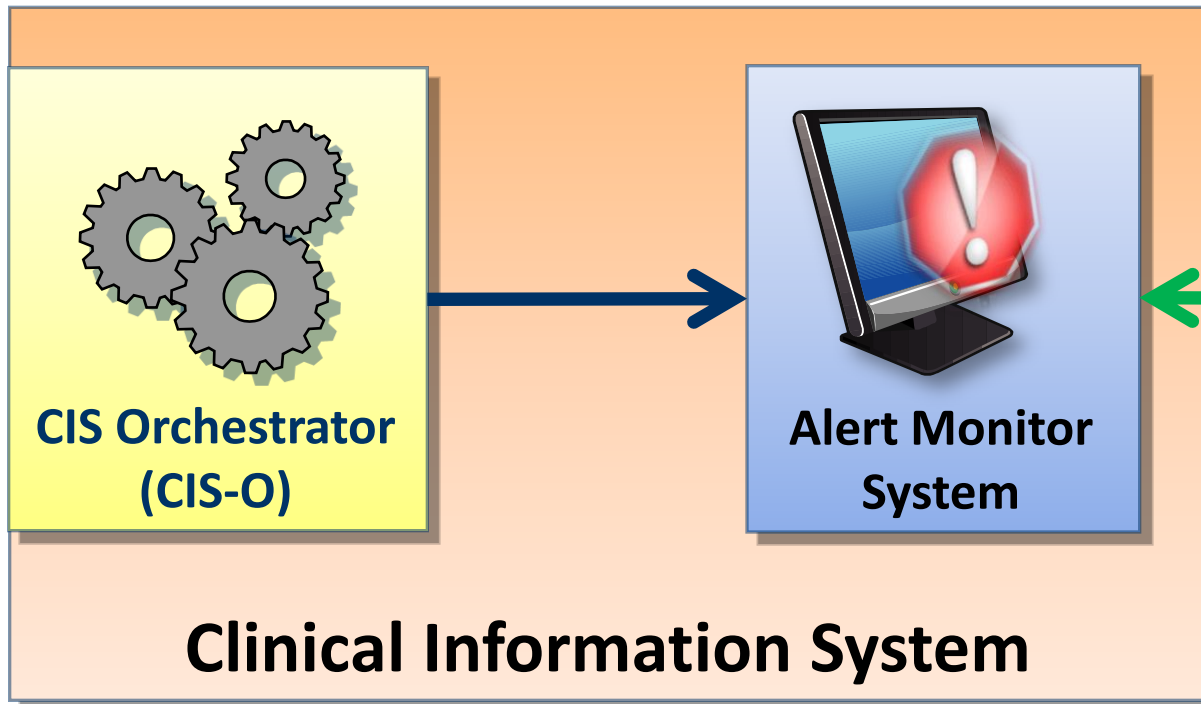
Nurse

Monitors & verifies alerts

Tying it Together: An Example Scenario



Outpatient monitoring system

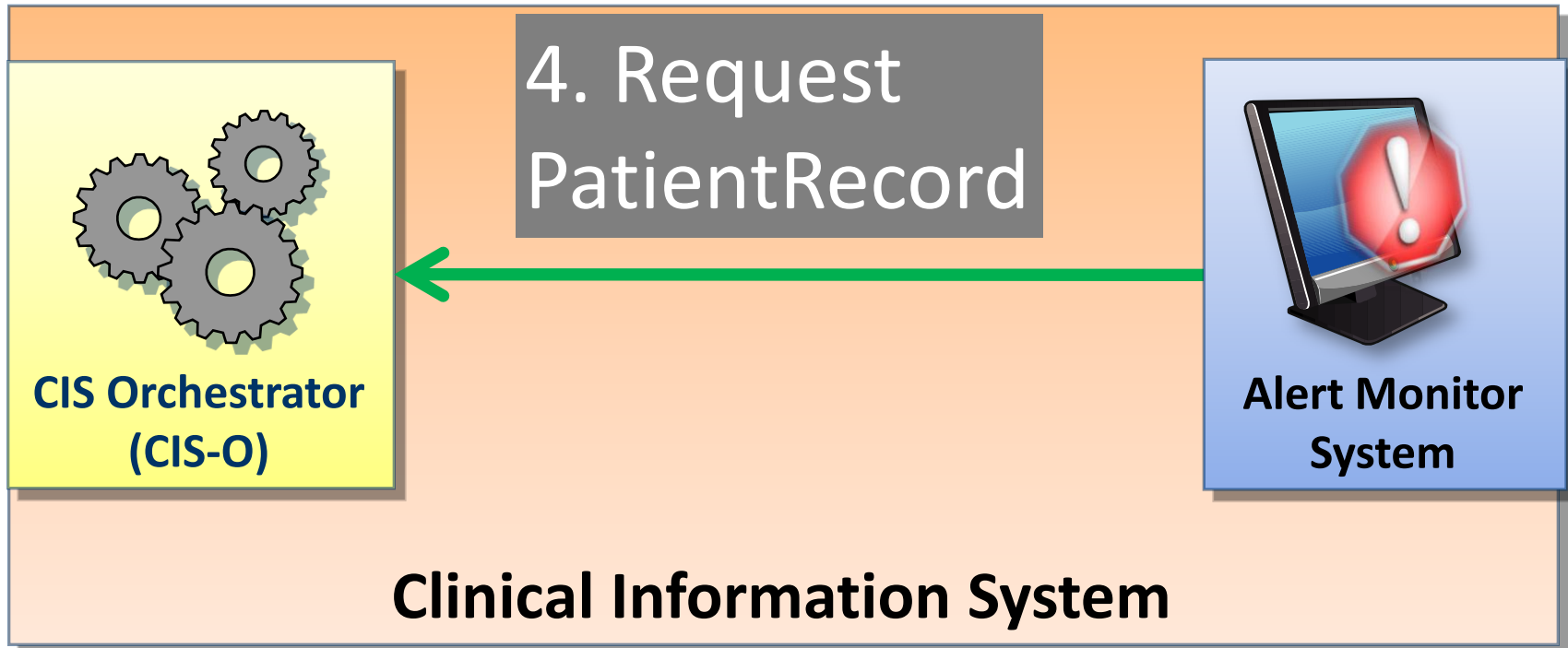


Monitors & verifies alerts

Tying it Together: An Example Scenario



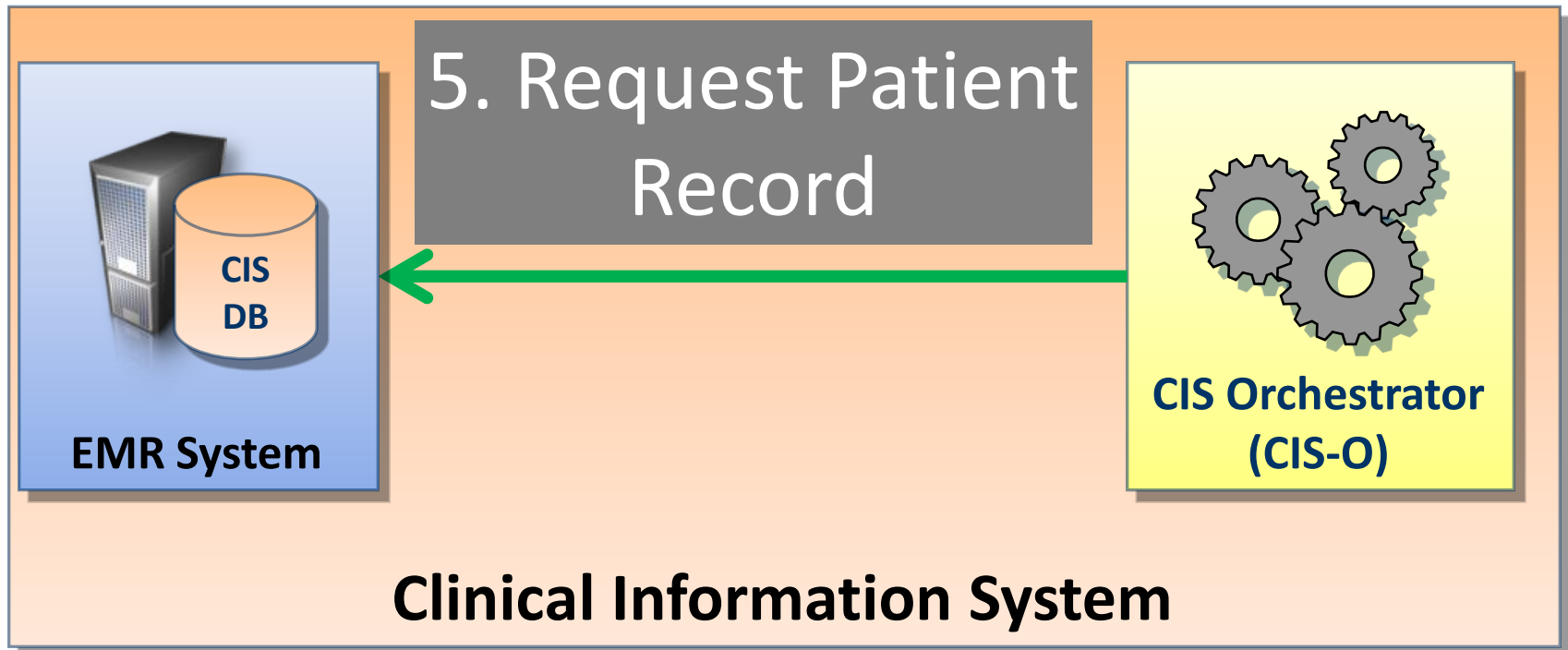
Outpatient monitoring system



Tying it Together: An Example Scenario



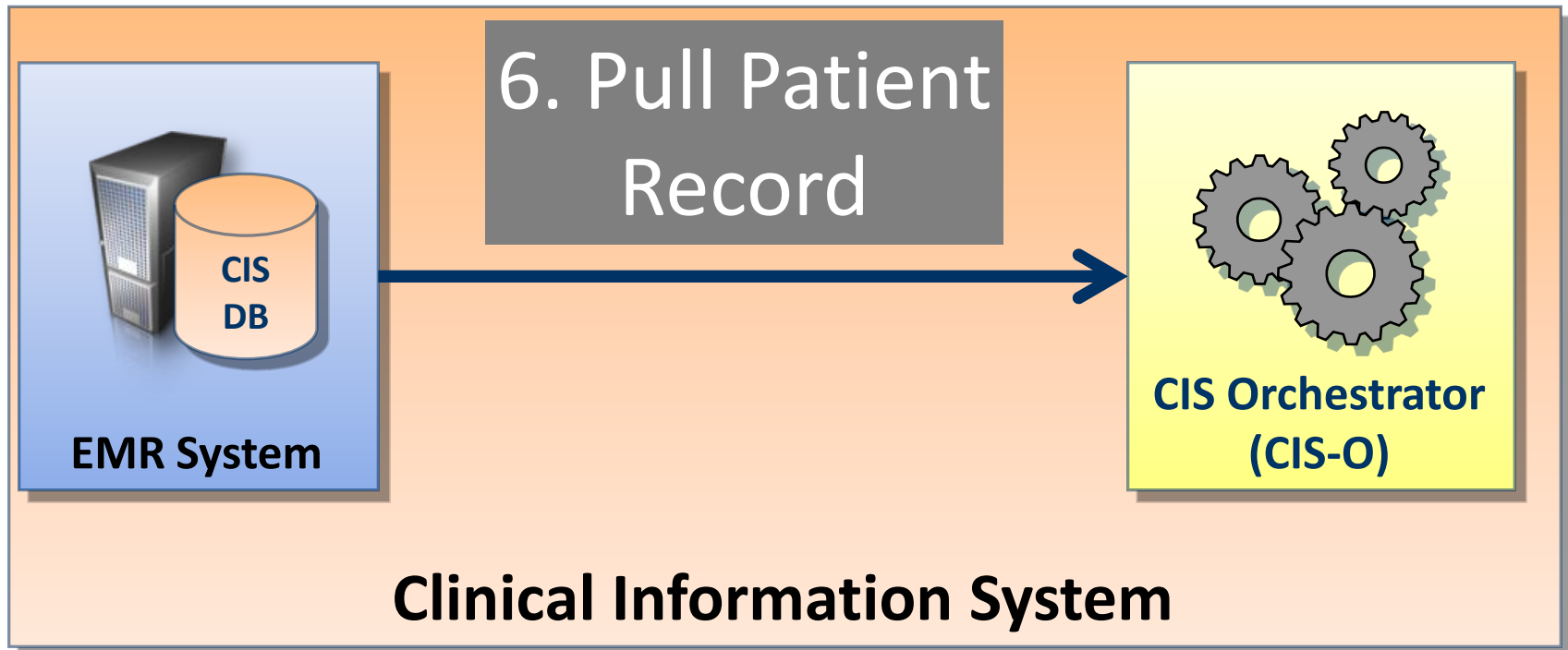
Outpatient monitoring system



Tying it Together: An Example Scenario



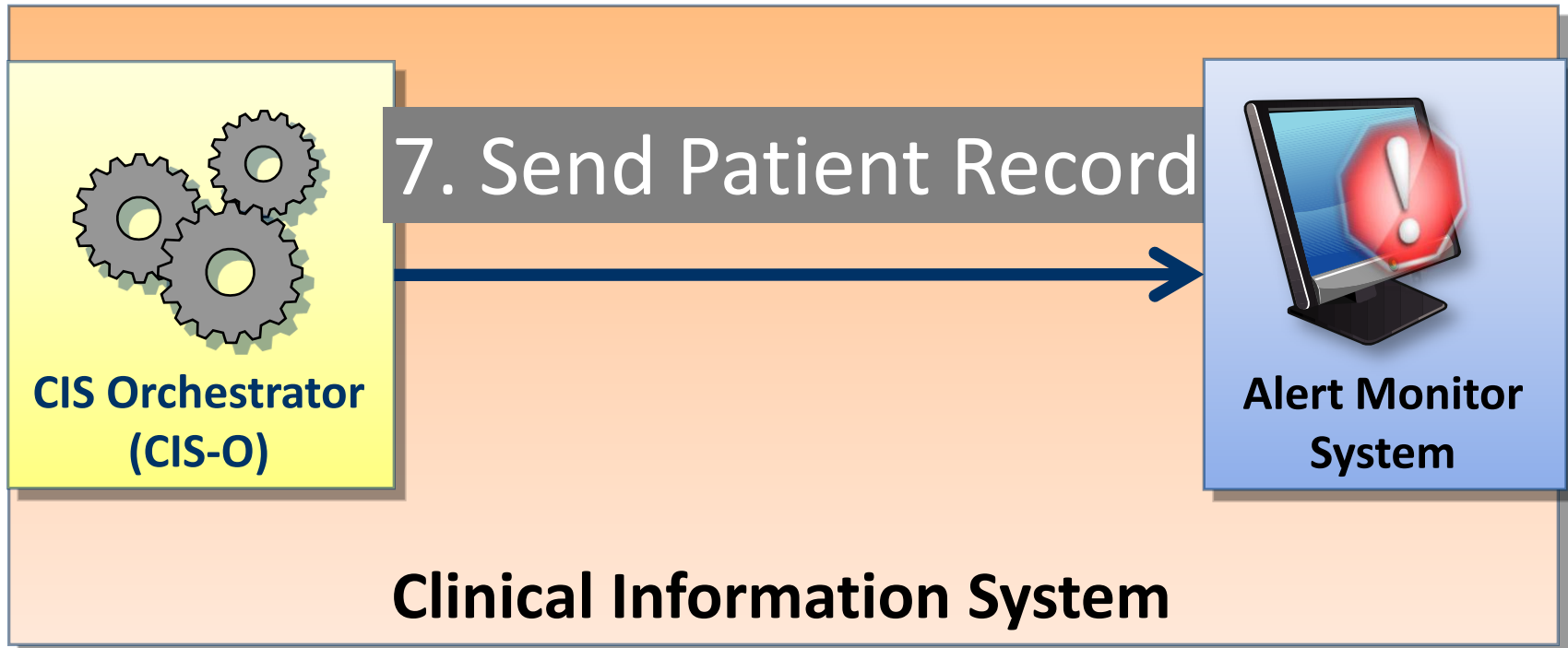
Outpatient monitoring system



Tying it Together: An Example Scenario



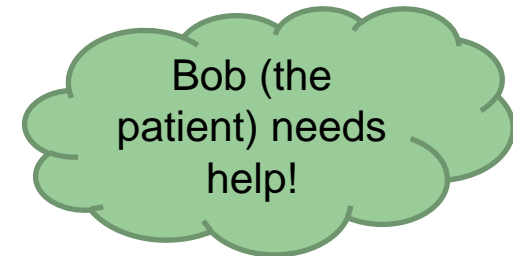
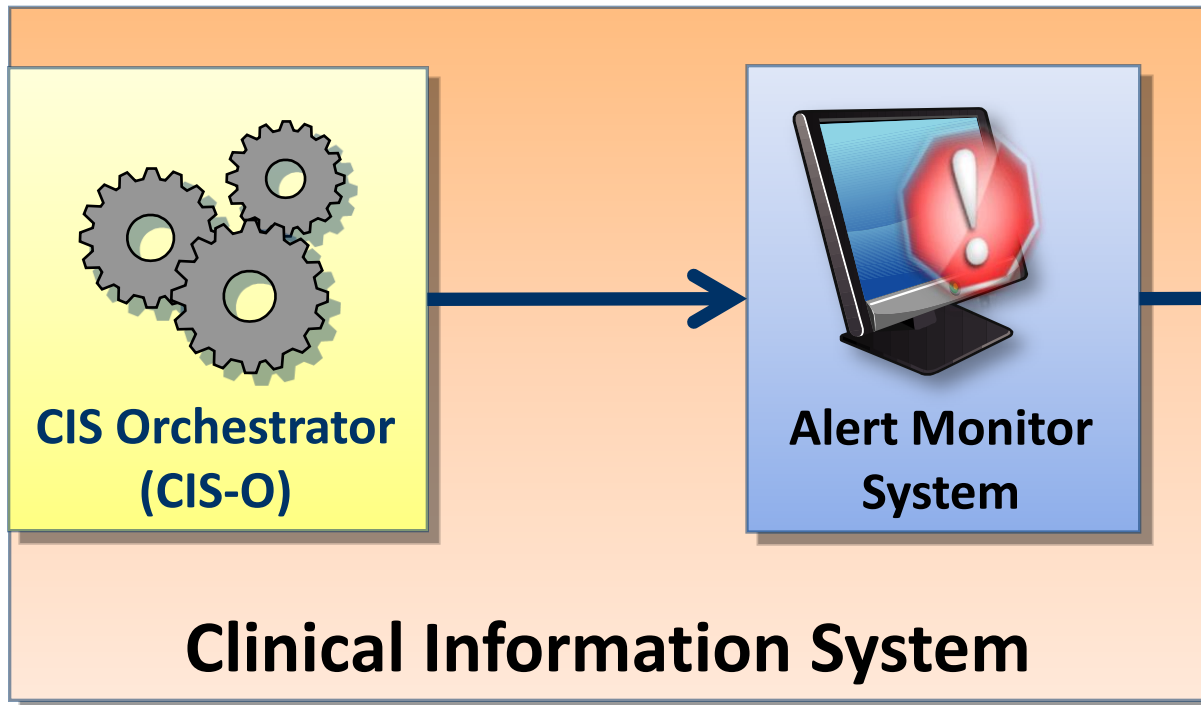
Outpatient monitoring system



Tying it Together: An Example Scenario



Outpatient monitoring system



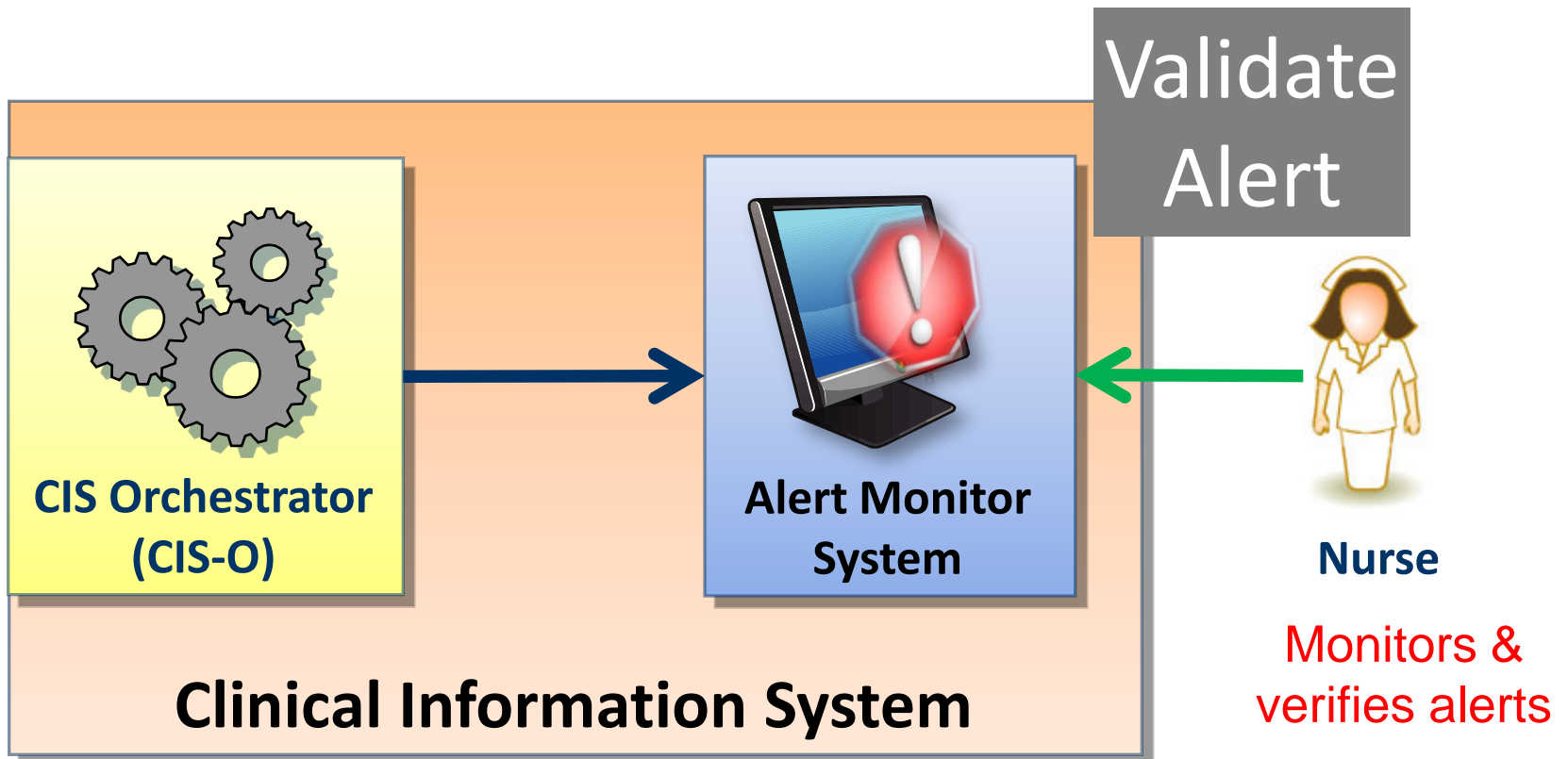
Nurse

Monitors & verifies alerts

Tying it Together: An Example Scenario



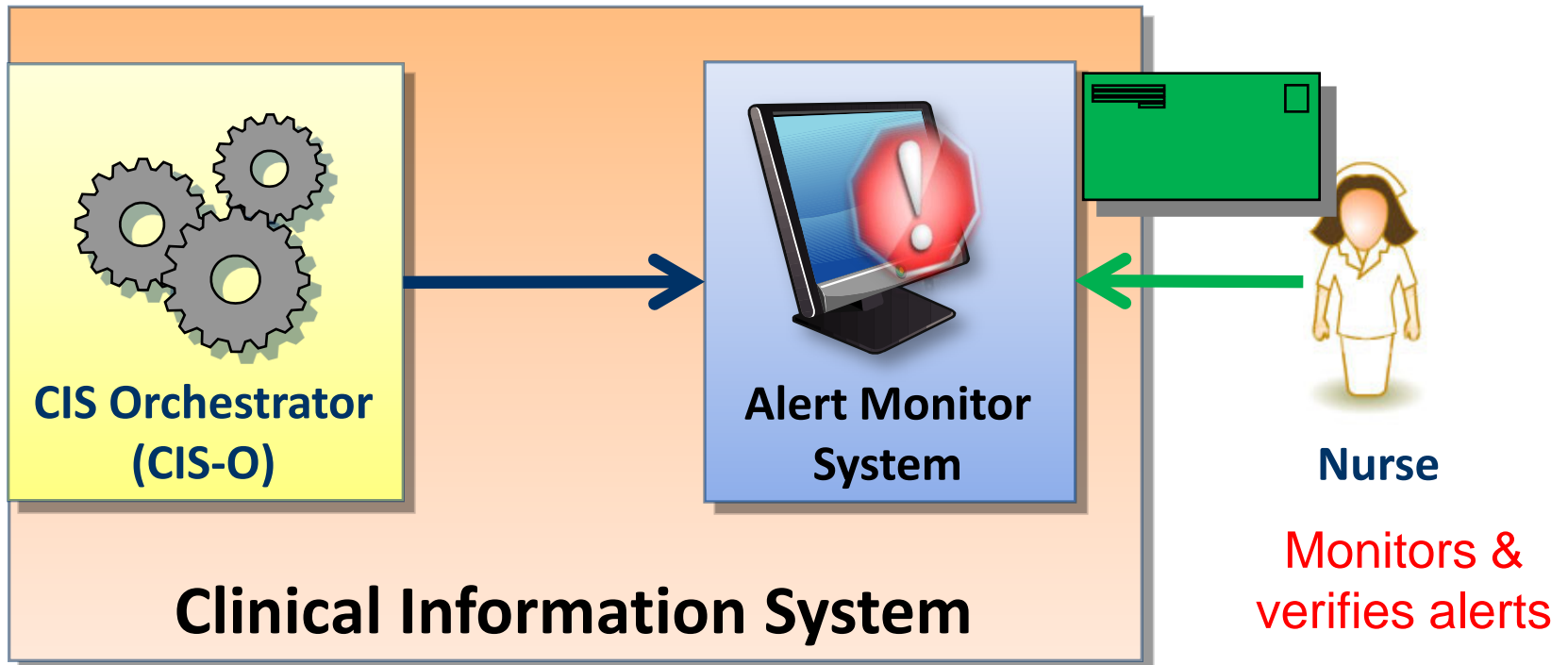
Outpatient monitoring system



Tying it Together: An Example Scenario



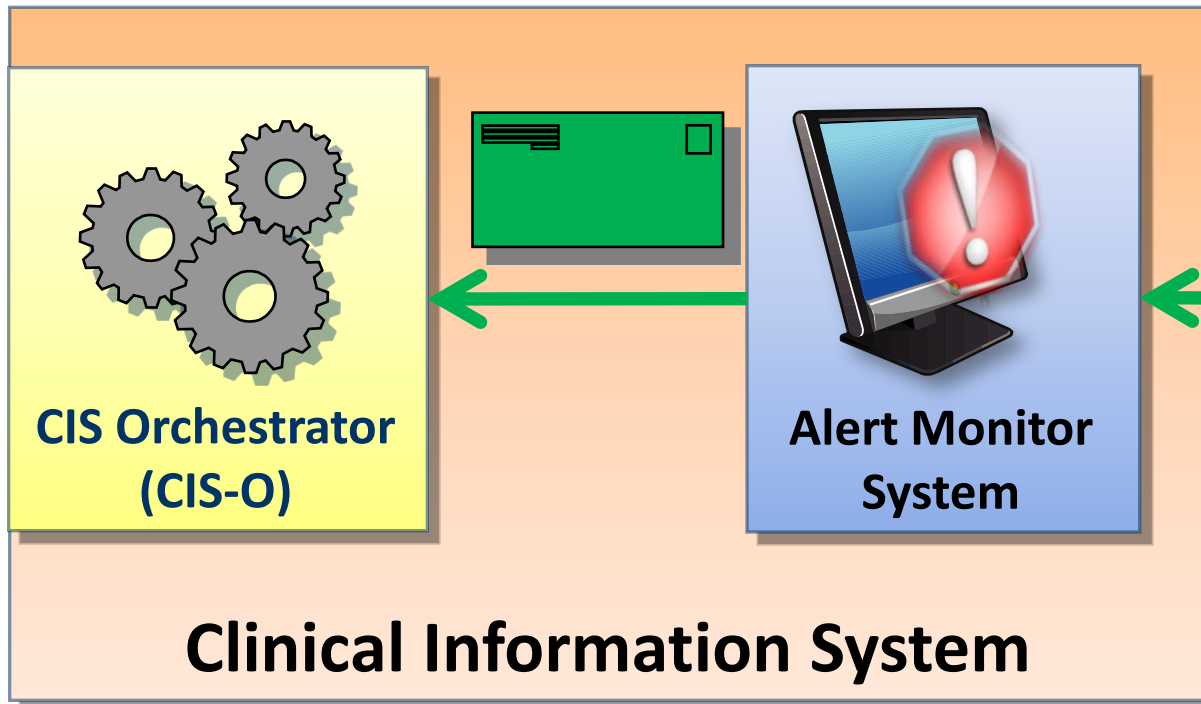
Outpatient monitoring system



Tying it Together: An Example Scenario



Outpatient monitoring system

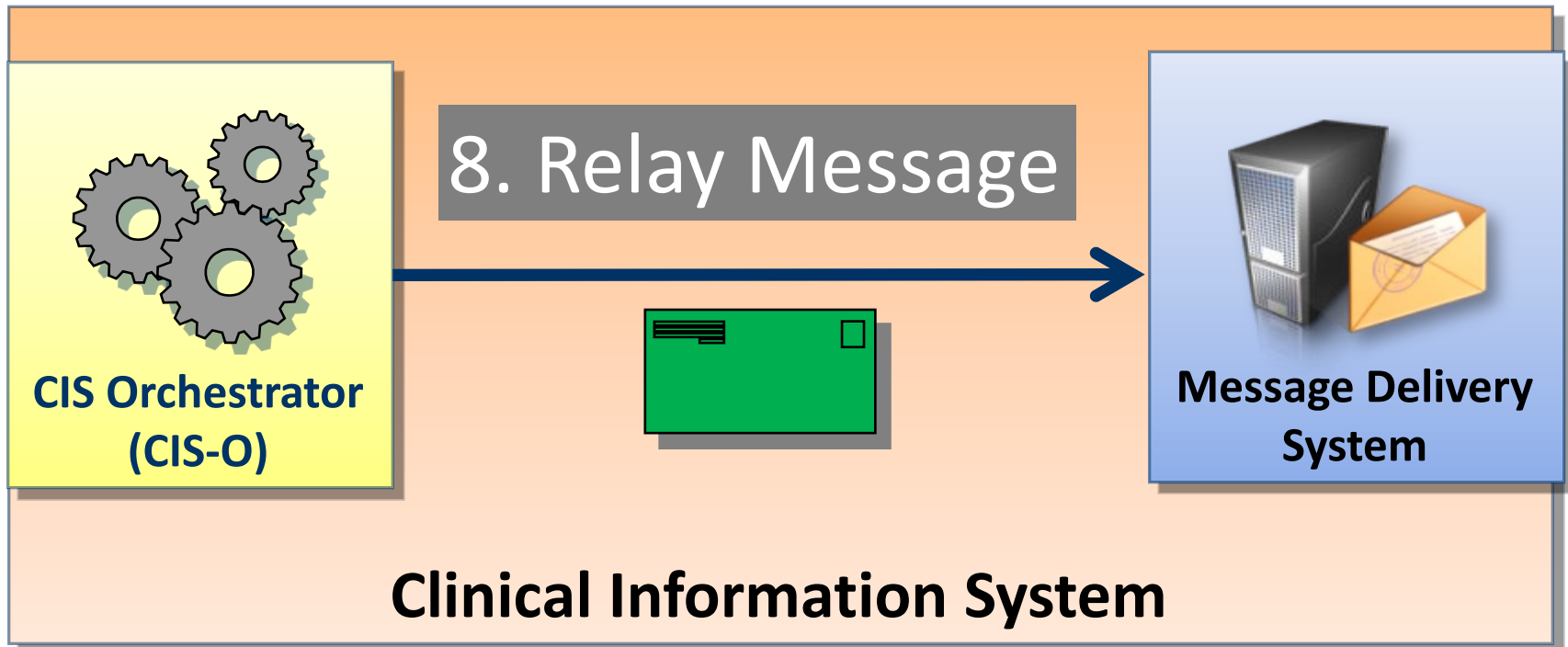


Monitors &
verifies alerts

Tying it Together: An Example Scenario



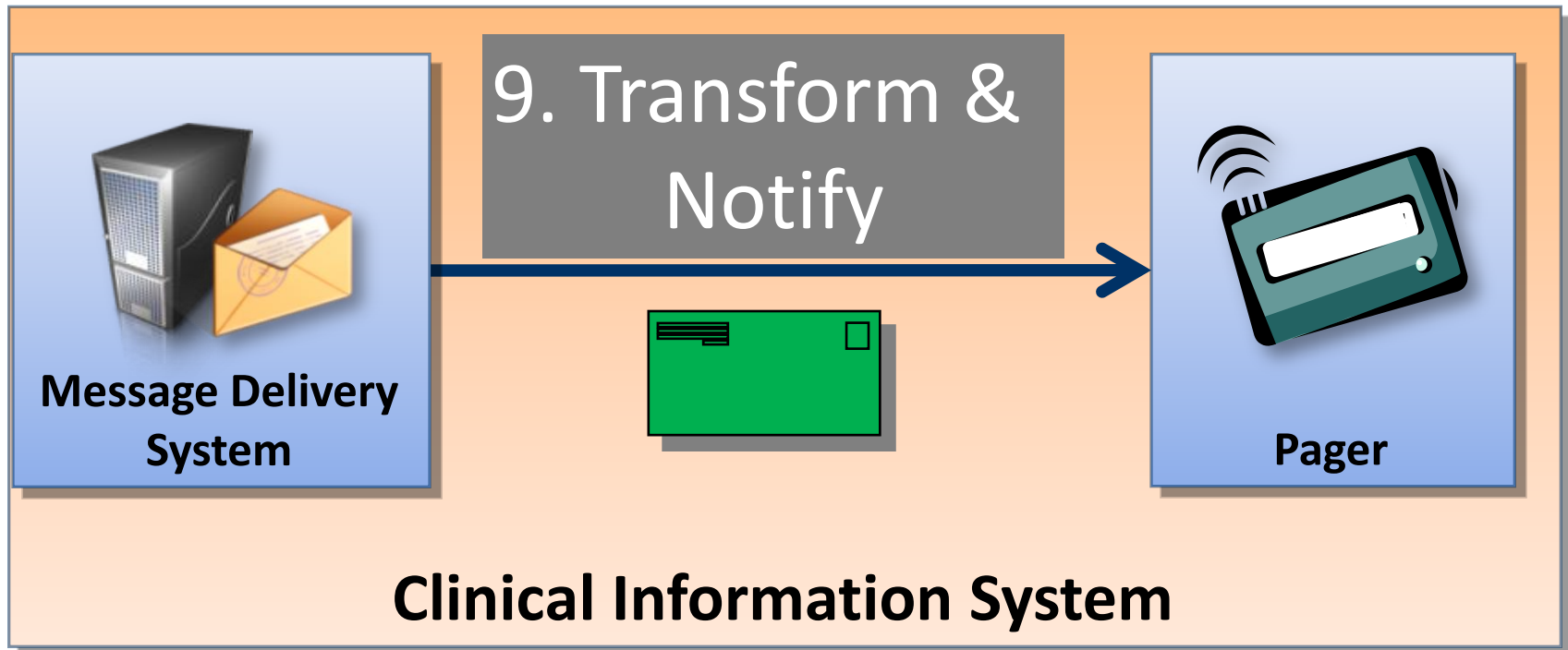
Outpatient monitoring system



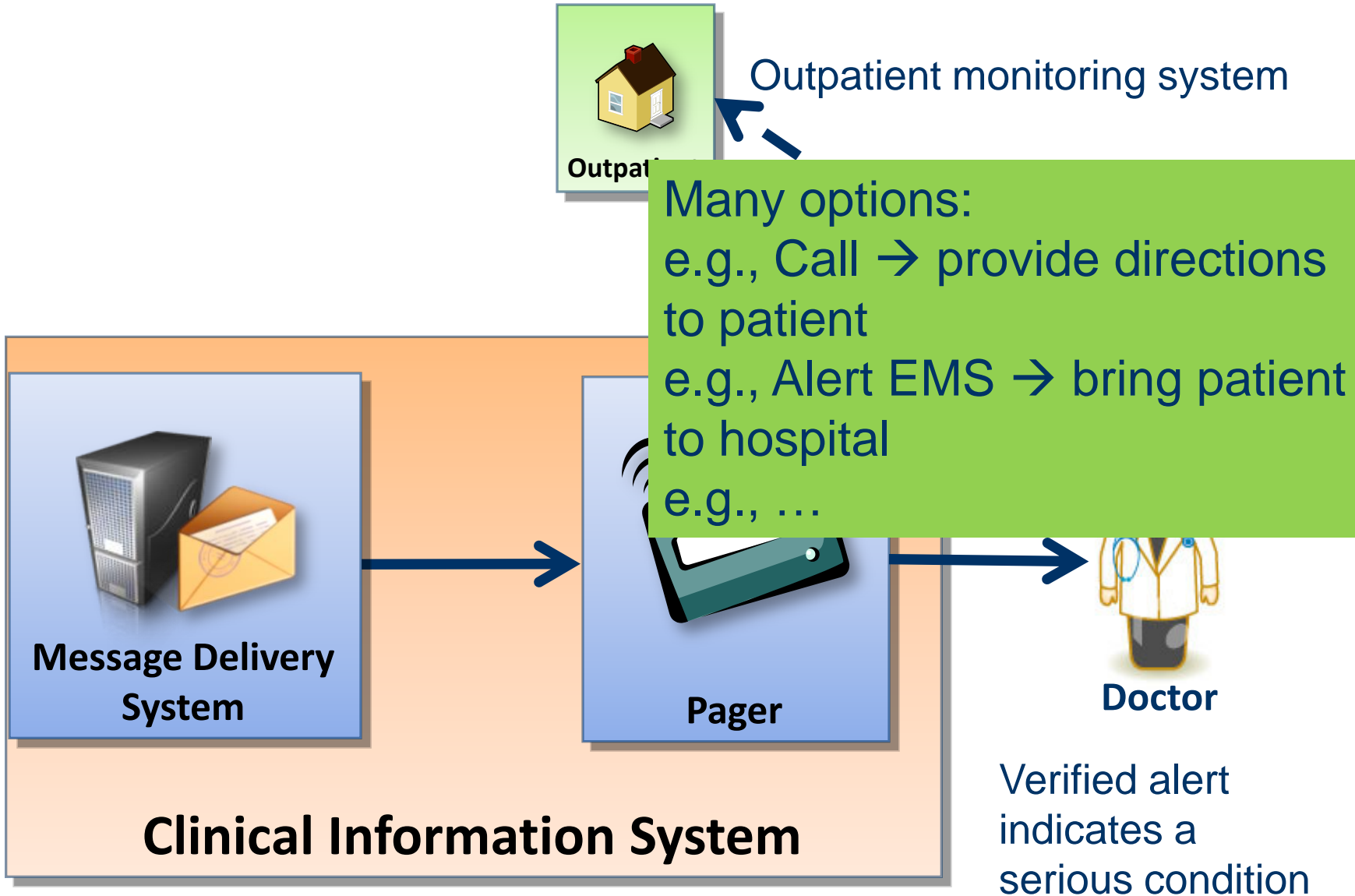
Tying it Together: An Example Scenario



Outpatient monitoring system



Tying it Together: An Example Scenario





Outpatient monitoring system
Wearable sensors, video capture, wireless networking
TRUST project (Berkeley, Cornell, Vanderbilt)

- Clinical information system services, workflows, policies, roles are all captured in the models
- The system is automatically generated and deployed

System

Pager

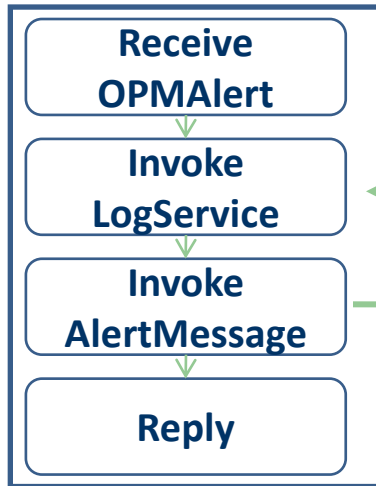
Clinical Information System

nd

ert
s

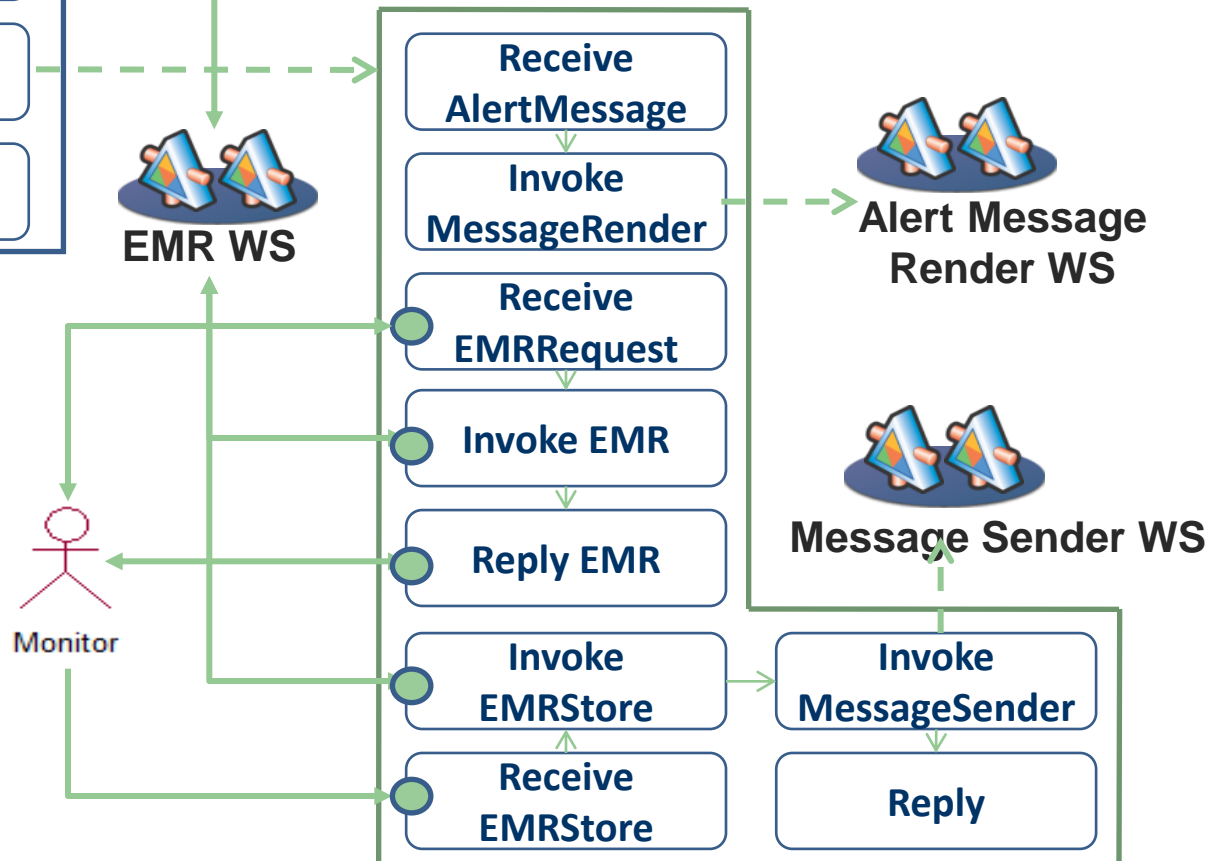
Example: A Little Deeper

OPMAAlertMain

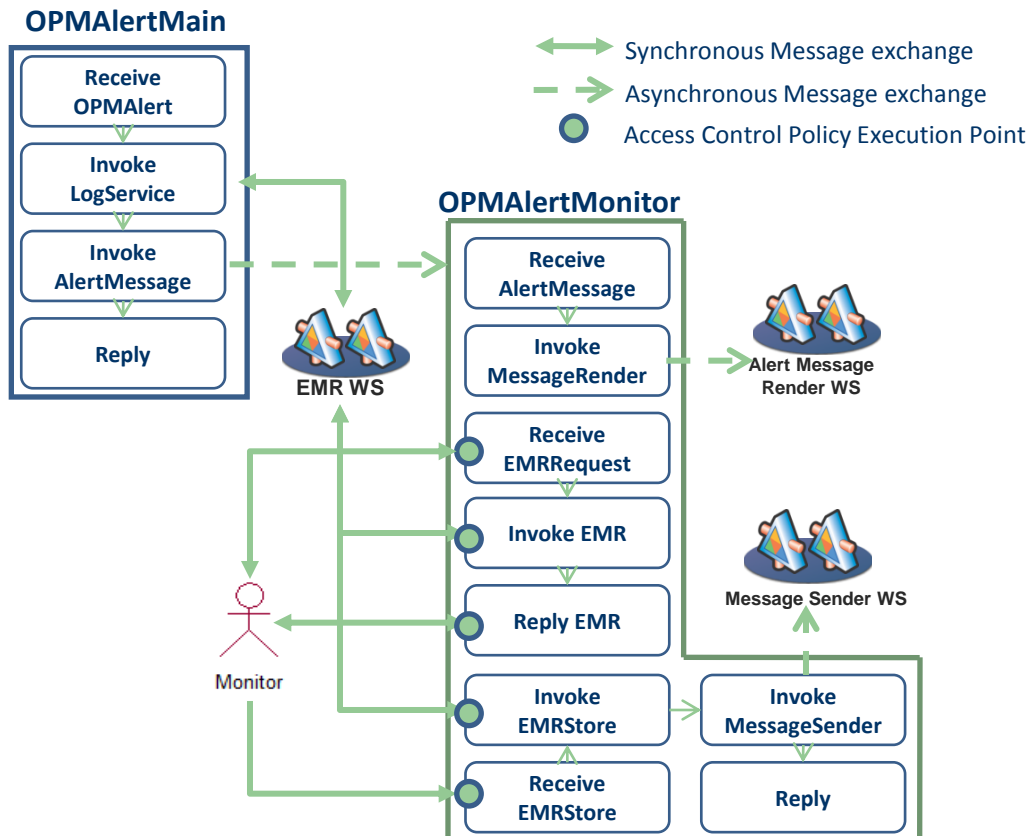


←→ Synchronous Message exchange
- - -> Asynchronous Message exchange
● Access Control Policy Execution Point

OPMAAlertMonitor



Example Scenario



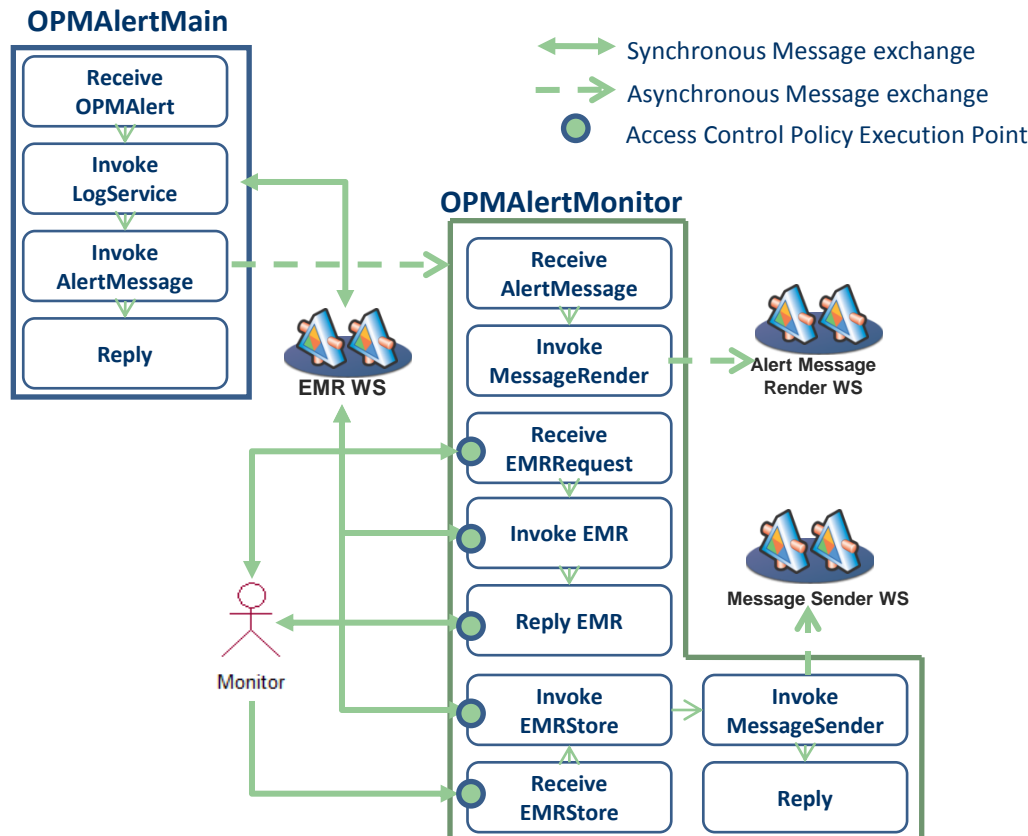
- When an anomaly is detected, the outpatient monitoring service issues an alert

- The clinical information system orchestrator (*CIS-O*) receives the alert message

- After logging alarm status in the EMR system, *CIS-O* sends the message to *Alert Monitor System* to render it on a monitoring station

- When the nurse checks the message → requests the patient's medical record to evaluate the situation

Example Scenario



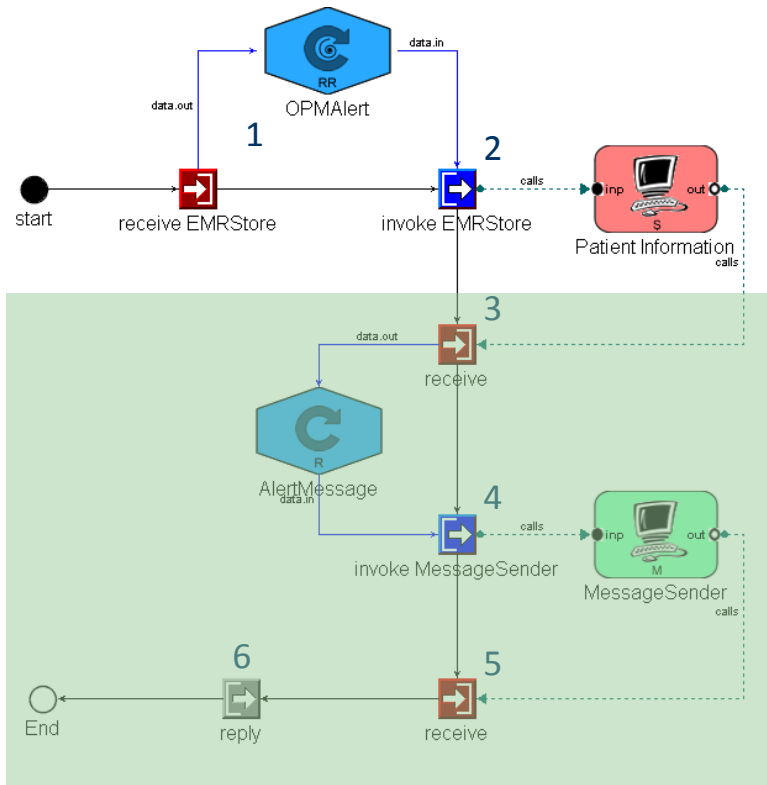
- Patient information includes medical history & contact information which can be used by the nurse to validate the alert

- If the alert is deemed important, she writes the status to the patient medical record

- Finally, *C/S-O* forwards the alert message to the designated doctors by using the *Message Delivery System*

- Otherwise, the alert message is stored in the EMR system and the process is terminated

Example: Sample Workflow Model



Workflow: OPMAAlertStore process

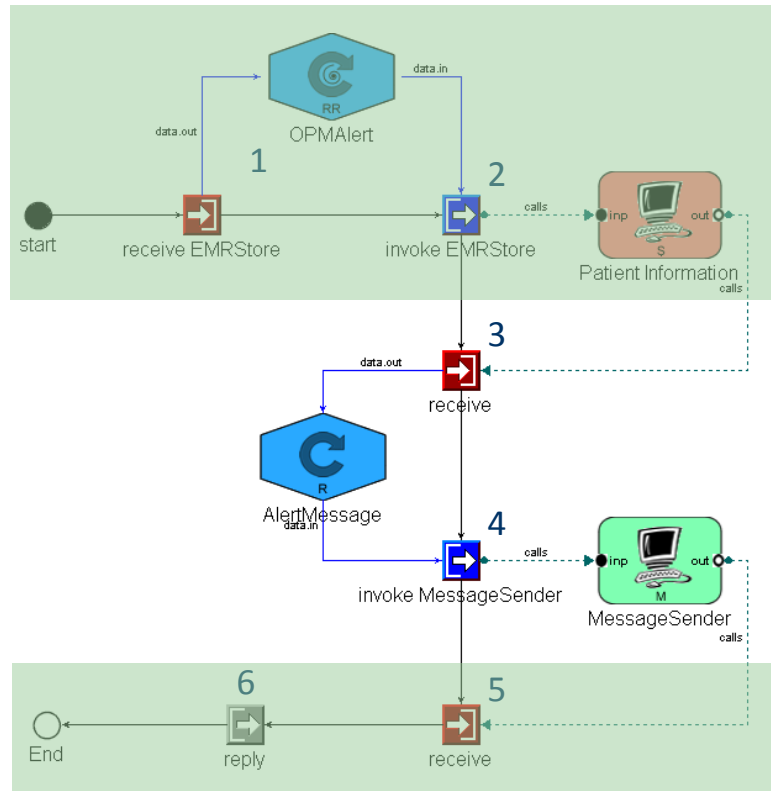
OPMAAlertStore Process

Goal: store the result of nurse's alert validation

Steps:

1. Alert status is assigned to the *OPMAAlert* data type
2. *Invoke EMRStore* activity invokes the *PatientInformation* web service
 - a) Store the validation results in the EMR System
 - b) Privacy policies applied when *invokeEMRStore* activity invokes *Patient Information* web service

Example: Sample Workflow Model



Workflow: OPMAAlertStore process

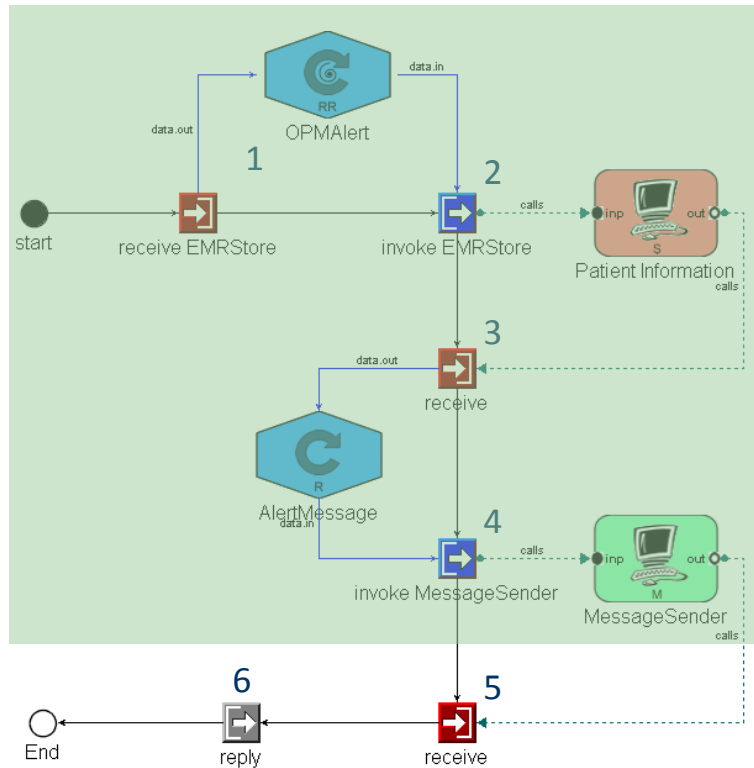
OPMAAlertStore Process

Goal: store the result of nurse's alert validation

Steps:

3. After the *receive* activity receives the acknowledge message from the web service, it assigns it to the *AlertMessage* variable
4. The *InvokeMessageSender* activity invokes the *MessageSender* web service to forward the alert message to the designated doctors via the Message Delivery System

Example: Sample Workflow Model



OPMAAlertStore Process

Goal: store the result of nurse's alert validation

Steps:

5. After the *MessageSender* web service is completed,
6. The *OPMAAlertstore* process returns.

Workflow: OPMAAlertStore process

Example: Policy Models

The screenshot shows the PMS software interface with a window titled "PMS - Root Folder - [NewMod...". The main area displays a modeling sheet with a vertical flow diagram. At the top is a green hexagon labeled "M". An arrow points down to a green rounded rectangle labeled "OPMAAlertMonitor" containing a computer icon. Another arrow points down to a second green hexagon labeled "M". A properties window at the bottom is open, showing the "Attributes" tab with the following content:

| Attributes | Preferences | Properties |
|-------------------------------|-------------|------------------|
| Name | | OPMAAlertRequest |
| Define type of the mt Request | | |

The screenshot shows the PMS software interface with a window titled "PMS - Root Folder - [OutPolicy - /Root Folder/NewModeli...". The main area displays a policy model editor with a list of rules and a diagram. The rules are:

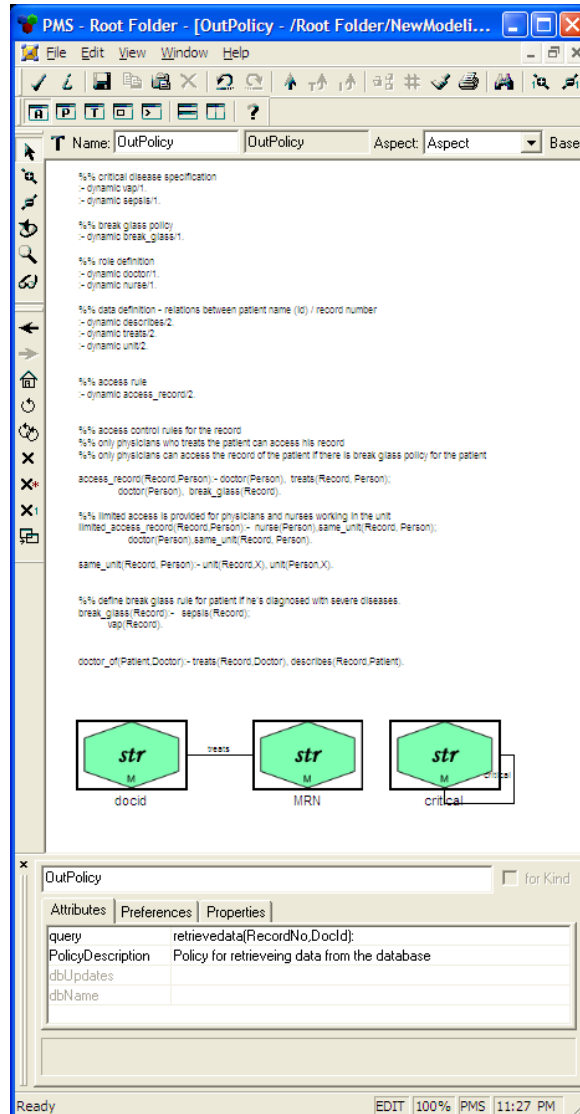
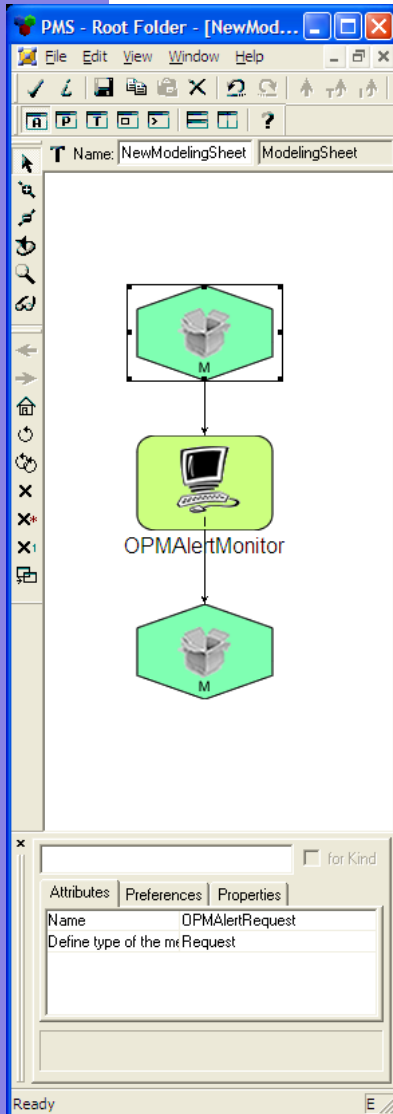
- % critical disease specification
 - dynamic vap:1
 - dynamic sepsis:1
- % break glass policy
 - dynamic break_glass:1
- % role definition
 - dynamic doctor:1
 - dynamic nurse:1
- % data definition - relations between patient name (id) / record number
 - dynamic describes:2
 - dynamic treats:2
 - dynamic unit:2
- % access rule
 - dynamic access_record:2
- % access control rules for the record
 - % only physicians who treats the patient can access his record
 - % only physicians can access the record of the patient if there is break glass policy for the patient
- access_record(Record,Person) - doctor(Person), treats(Record,Person);
doctor(Person), break_glass(Record)
- % limited access is provided for physicians and nurses working in the unit
 - limited_access_record(Record,Person) - nurse(Person).same_unit(Record,Person);
doctor(Person).same_unit(Record,Person)
- same_unit(Record,Person) - unit(Record,X), unit(Person,X)
- % define break glass rule for patient if he's diagnosed with severe diseases.
 - break_glass(Record) - sepsis(Record);
vap(Record)
- doctor_of(Patient,Doctor) - treats(Record,Doctor), describes(Record,Patient)

The diagram at the bottom shows three green hexagons labeled "str" with "M" below them. The first is labeled "docid", the second "MRN", and the third "critical". A line labeled "treats" connects "docid" to "MRN". A line labeled "describes" connects "MRN" to "critical".

A properties window at the bottom is open, showing the "Attributes" tab with the following content:

| Attributes | Preferences | Properties |
|-------------------|-------------|-----------------------------------------------|
| query | | retrieveData(RecordNo,DocId); |
| PolicyDescription | | Policy for retrieveing data from the database |
| dbUpdates | | |
| dbName | | |

Example: Policy Models



Policies Defined for Scenario

- Only medical staff is allowed to access alert messages
- Only primary care physicians are allowed to access patient's medical record
- The nurse is allowed to access the records of patients monitored by the OPM system
- Medical staff is allowed to access patient's record in emergency situation triggering the Break Glass policy

- Policy description includes
 - Definition of incoming & outgoing data
 - Evaluation point
 - Obligations
 - Additional datasets for policy evaluation
- Model contains information required to generate the policy:
 - Query evaluated to determine access rights
 - Attribute relations used for policy evaluation
 - Textual policy description

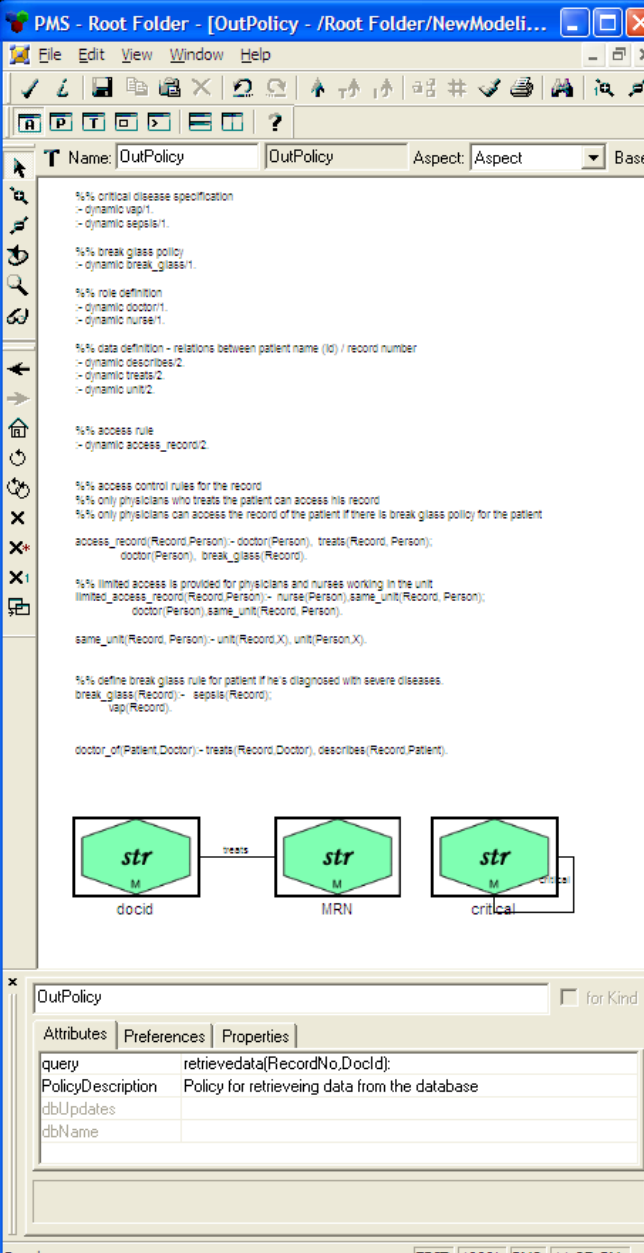
Example: Policy Models

- Example query:
 - *retrieveData(PatientID, staffID)* after the service has been executed
 - Use a redefined set of predicates and attribute relations

(is_critical() , treats(staffID,MRN))

- These are generated from
 - incoming data
 - outgoing data

by the Policy Enforcement Point (PEP)



The screenshot shows the PMS interface with the following components:

- Window Title:** PMS - Root Folder - [OutPolicy - /Root Folder/NewModeli...
- Menu:** File, Edit, View, Window, Help
- Toolbar:** Standard application toolbar with icons for file operations and navigation.
- Policy Editor:**
 - Name:** OutPolicy
 - Aspect:** Aspect
 - Base:** Aspect
 - Code:**

```

%% critical disease specification
- dynamic vap/1.
- dynamic sepsis/1.

%% break glass policy
- dynamic break_glass/1.

%% role definition
- dynamic doctor/1.
- dynamic nurse/1.

%% data definition - relations between patient name (ID) / record number
- dynamic describes/2.
- dynamic treats/2.
- dynamic unit/2.

%% access rule
- dynamic access_record/2.

%% access control rules for the record
%% only physicians who treats the patient can access his record
%% only physicians can access the record of the patient if there is break glass policy for the patient
access_record(Record,Person):- doctor(Person), treats(Record, Person);
doctor(Person), break_glass(Record).

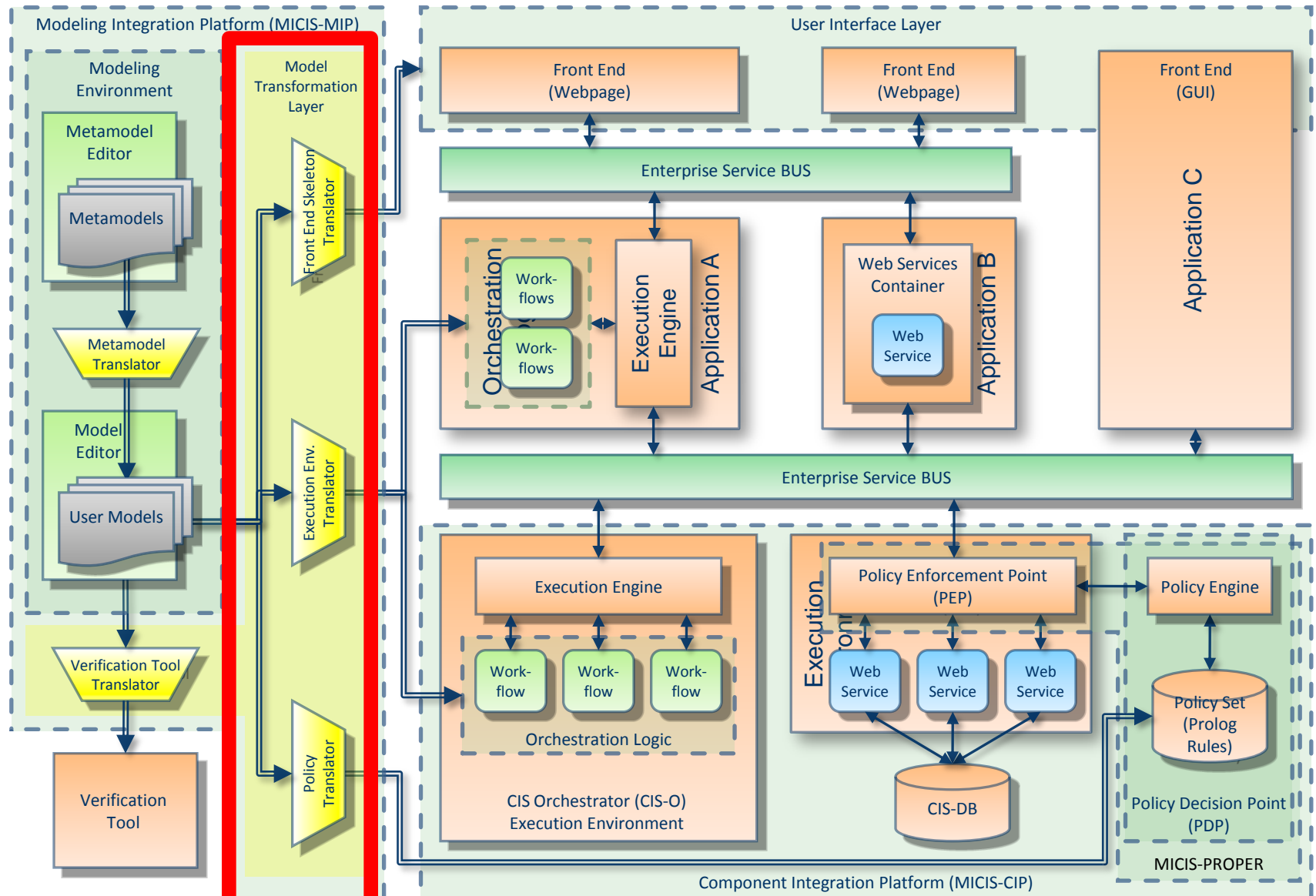
%% limited access is provided for physicians and nurses working in the unit
limited_access_record(Record,Person):- nurse(Person),same_unit(Record, Person);
doctor(Person),same_unit(Record, Person).

same_unit(Record, Person):- unit(Record,X), unit(Person,X).

%% define break glass rule for patient if he's diagnosed with severe diseases.
break_glass(Record):- sepsis(Record);
vap(Record).

doctor_of(Patient,Doctor):- treats(Record,Doctor), describes(Record,Patient).
          
```
- Diagram:** A graph showing three nodes labeled 'str' (green hexagons) with 'M' below them. The nodes are labeled 'docid', 'MRN', and 'critical'. An edge labeled 'treats' connects 'docid' to 'MRN'. An edge labeled 'critical' connects 'MRN' to 'critical'.
- Properties Panel:**
 - Attributes:** query, PolicyDescription, dbUpdates, dbName
 - Values:** retrieveData(RecordNo,DocId);, Policy for retrieveing data from the database, ,

Magic: *Transform* → Code



Code Generation

```
UltraEdit-32 - [C:\Documents and Settings\ylee\Desktop\policy-out.xml]
<?xml version="1.0" encoding="UTF-8"?>
<PolicyList>
  <PolicyDescription>
    <methodName>retrievedata</methodName>
    <query>retrievedata (RecordNo,DocId) </query>
    <requestFields>RecordNo</requestFields>
    <replyFields>critical; DocId</replyFields>
    <relations>is_critical (critical) </relations>
    <inPolicy>true</inPolicy>
    <policyDbName></policyDbName>
    <dbUpdates></dbUpdates>
  </PolicyDescription>
</PolicyList>
```

```
UltraEdit-32 - [C:\Documents and Settings\ylee\Desktop\policies.pl]
policy-out.xml policies.pl

:- dynamic access_record/2.

%% access control rules for the record
%% only physicians who treats the patient can access his record
%% only physicians can access the record of the patient if there is break

retrievedata(RecordNo,DocId):- treats(RecordNo, DocId);
                           break_glass(RecordNo) .

%% limited access is provided for physicians and nurses working in the un

limited_access_record(RecordNo,DocId):- same_unit(RecordNo, DocId) .

same_unit(RecordNo, Person):- unit(RecordNo,X), unit(Person,X) .

%% define break glass rule for patient if he's diagnosed with severe dise.
break_glass(RecordNo):- critical(RecordNo,X), X>0.
```

Policy Description

Policy Document

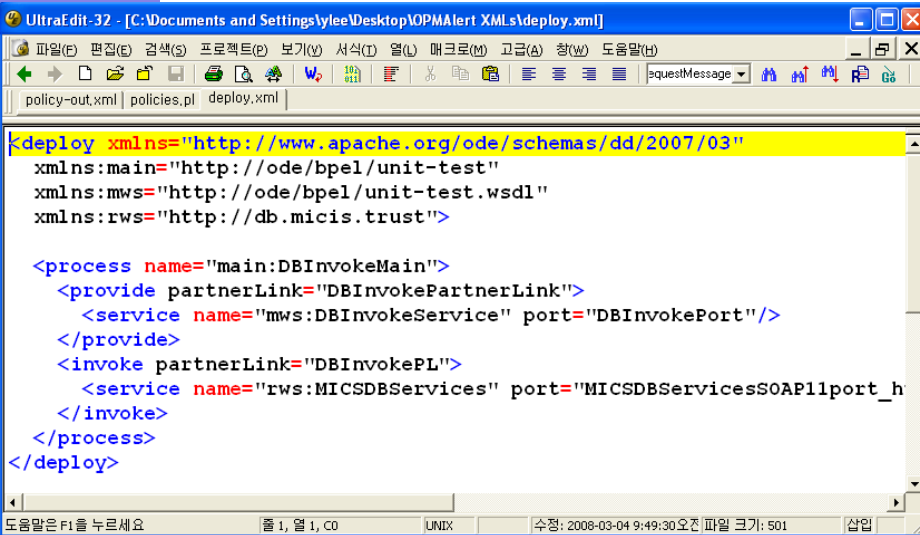
**Policy
Translator**



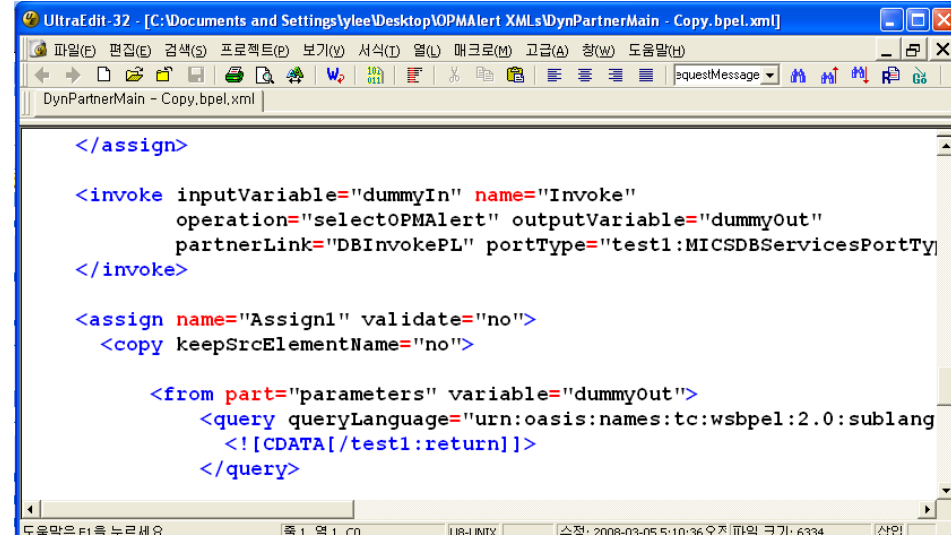
Execution Environment Translator

Deploy.xml

**BPELDocument
(OPMAAlertMain Process)**



```
<?xml version="1.0" encoding="UTF-8" ?>
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
xmlns:main="http://ode/bpel/unit-test"
xmlns:mws="http://ode/bpel/unit-test.wsdl"
xmlns:rws="http://db.micis.trust">
  <process name="main:DBInvokeMain">
    <provide partnerLink="DBInvokePartnerLink">
      <service name="mws:DBInvokeService" port="DBInvokePort"/>
    </provide>
    <invoke partnerLink="DBInvokePL">
      <service name="rws:MICSDBServices" port="MICSDBServicesSOAP11port_h">
      </service>
    </invoke>
  </process>
</deploy>
```



```
</assign>
<invoke inputVariable="dummyIn" name="Invoke"
operation="selectOPMAAlert" outputVariable="dummyOut"
partnerLink="DBInvokePL" portType="test1:MICSDBServicesPortTy">
</invoke>
<assign name="Assign1" validate="no">
  <copy keepSrcElementName="no">
    <from part="parameters" variable="dummyOut">
      <query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang
      <![CDATA[/test1:return]]>
    </query>
  </from>
</assign>
```


Code Generation

Execution Environment Translator

**WSDL for OPMAAlertMain
Process**

**WSDL for Patient
Information Web Service**

```
UltraEdit-32 - [C:\Documents and Settings\ylee\Desktop\OPMAAlert XMLs\Main.wsdl]
파일(F) 편집(E) 검색(S) 프로젝트(P) 보기(V) 서식(T) 열(L) 매크로(M) 고급(A) 창(W) 도움말(H)
DynPartnerMain - Copy.bpel.xml | Main.wsdl

<wsdl:part name="ContextPayload" element="tns:dummy"/>
</wsdl:message>

<wsdl:portType name="DBInvokePortType">
  <wsdl:operation name="execute">
    <wsdl:input message="tns:ReqeustMessage" name="request"/>
    <wsdl:output message="tns:ResultMessage" name="result"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="DBInvokeBinding" type="tns:DBInvokePortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/
  <wsdl:operation name="execute">
    <soap:operation soapAction="" style="document"/>
  </wsdl:operation>
</wsdl:binding>
</wsdl:service>

도움말은 F1을 누르세요 | 줄 1, 열 1, CO | U8-UNIX | 수정: 2008-03-04 7:14:02오전 파일 크기: 8664 | 삽입
```

```
UltraEdit-32 - [C:\Documents and Settings\ylee\Desktop\OPMAAlert XMLs\MICSDBServices.wsdl]
파일(F) 편집(E) 검색(S) 프로젝트(P) 보기(V) 서식(T) 열(L) 매크로(M) 고급(A) 창(W) 도움말(H)
DynPartnerMain - Copy.bpel.xml | Main.wsdl | MICSDBServices.wsdl

<wsdl:input message="ns1:updateAlertStateRequest" wsaw:Action="urn:
<wsdl:output message="ns1:updateAlertStateResponse"
  wsaw:Action="urn:updateAlertStateResponse"/>
<wsdl:fault message="ns1:Exception" name="Exception"
  wsaw:Action="urn:updateAlertStateException"/>
</wsdl:operation>
<wsdl:operation name="selectOPMAAlert">
  <wsdl:input message="ns1:selectOPMAAlertRequest" wsaw:Action="urn
  <wsdl:output message="ns1:selectOPMAAlertResponse" wsaw:Action="u
  <wsdl:fault message="ns1:Exception" name="Exception"
    wsaw:Action="urn:selectOPMAAlertException"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="MICSDBServicesSOAP11Binding" type="ns1:MICSDBServi

도움말은 F1을 누르세요 | 줄 1, 열 1, CO | U8-DO5 | 수정: 2008-03-04 9:40:38오전 파일 크기: 20766 | 삽입
```

- **Experimental Platform for EMR research**
 - Helping to solve privacy and security challenges of EMR *systems* applications
 - Usable for the integration, testing and evaluation of new technologies
- **Ongoing technology transition: Experimental Sepsis Management System for ICUs:**
 - Sepsis management protocol is formally defined: evidence-based medicine
 - Sepsis Management System is mapped on SOA platform
 - Model-Integrated systems approach

Acknowledgements

- NSF TRUST (CCF-0424422)

- Research Team



Yonghwan Lee



Akos Ledeczi, Ph.D.



Janos Mathe



Brad Malin, Ph.D.



Jan Werner



Janos Sztipanovits, Ph.D.