

The SENSORIA Development Environment

CASE Tool for SOA Development

SENSORIA EU FP6 project
19 partners from 7 countries
4 years, 4 M EUR



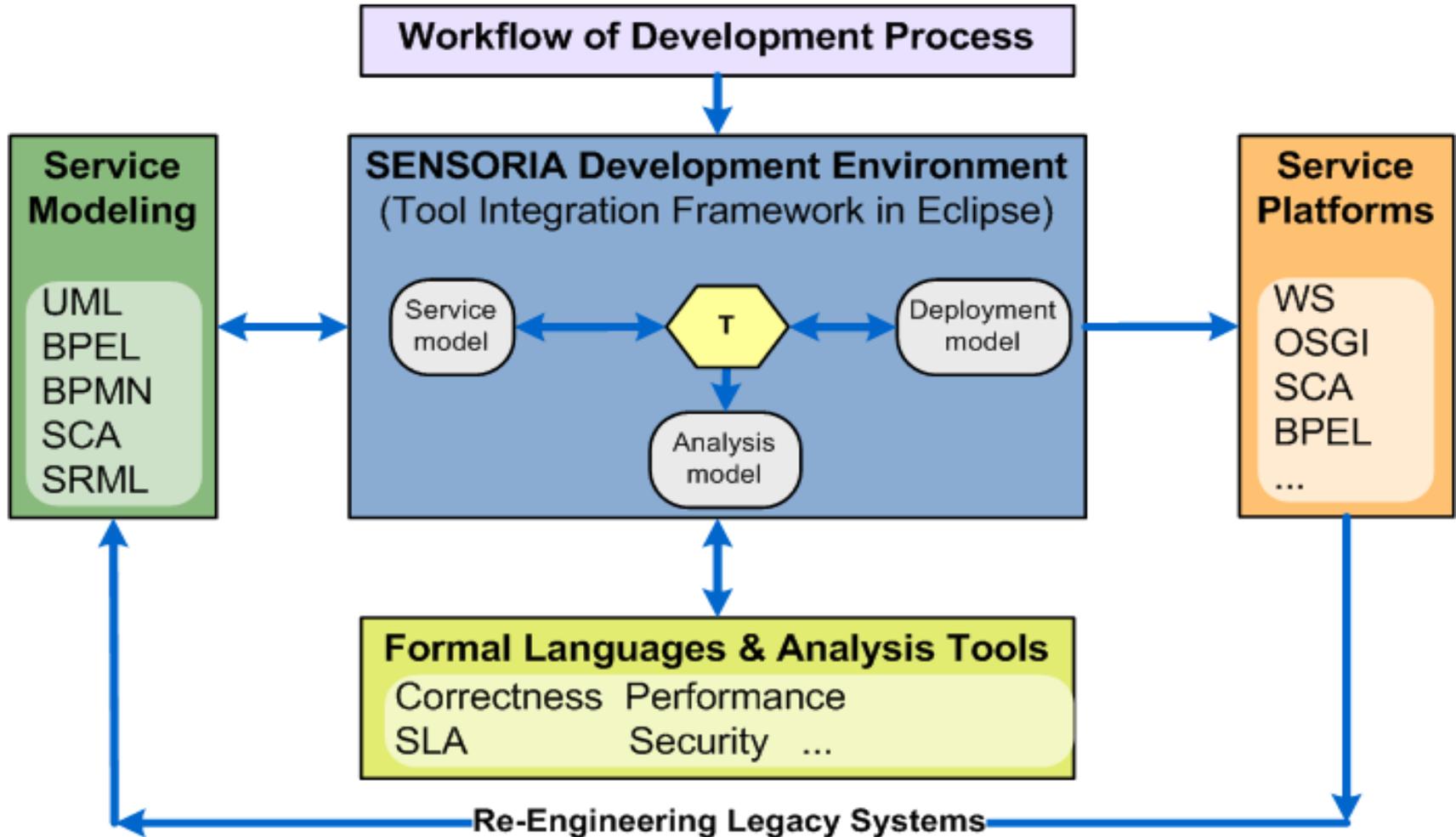
Coordinator: Prof. Dr. Martin Wirsing, Ludwig-Maximilians-Universität München, Germany
Università di Trento | University of Leicester | Warsaw University | TU Denmark at Lyngby | Università di Pisa
Università di Firenze | Università di Bologna | ISTI Pisa | Universidade de Lisboa | University of Edinburgh
ATX Software | SATelecom Italia Lab | Imperial College London | Cirquent GnbH (FAST GmbH) | Budapest University of
Technology and Economics | S&N AG | University College London | Politecnico di Milano

István Ráth
rath@mit.bme.hu

Context

- Service-Oriented Computing
 - Create new services
 - By composing existing functionality
 - With „Loosely-coupled” interaction
 - >Resilient to changing (business) environment
- Envisaged market impact
 - **Time-to-market of service integration** can be reduced
 - Increased **quality of service**
(security, response time, throughput, availability....)
 - Portability to existing platforms to **maintain profitable services**
- Challenges
 - **Specification and querying** of services?
 - **Correctness and consistency** of service composition?
 - Continuous operation in **changing environment** (service outages)?
 - Design for **justifiable SLA-compliance** (security, performance)?

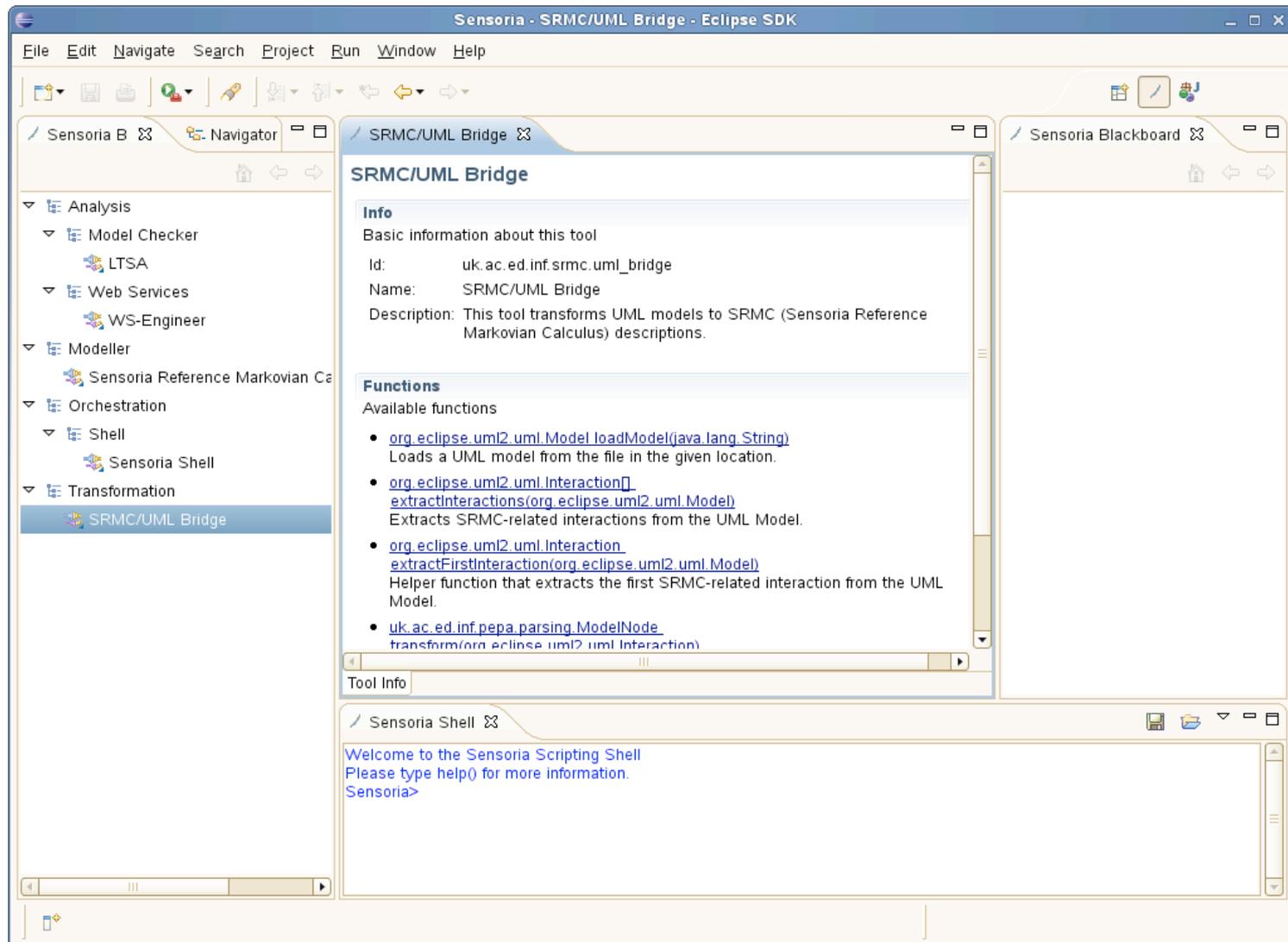
The SENSORIA Approach



The SENSORIA Development Environment (SDE)

- Objective of the SDE
 - Provide a service-oriented platform for tool integration
- Benefits
 - Discover & install SOA development tools
 - Orchestrate & compose tools as services
 - A homogeneous platform for the service-based toolchain
- SDE is based on SOA
 - OSGi for services
 - Eclipse for UI

The SDE UI



The SDE UI

Tool Browser
Lists available tools, ordered by category

Tools can be discovered, downloaded and installed via Eclipse update sites.

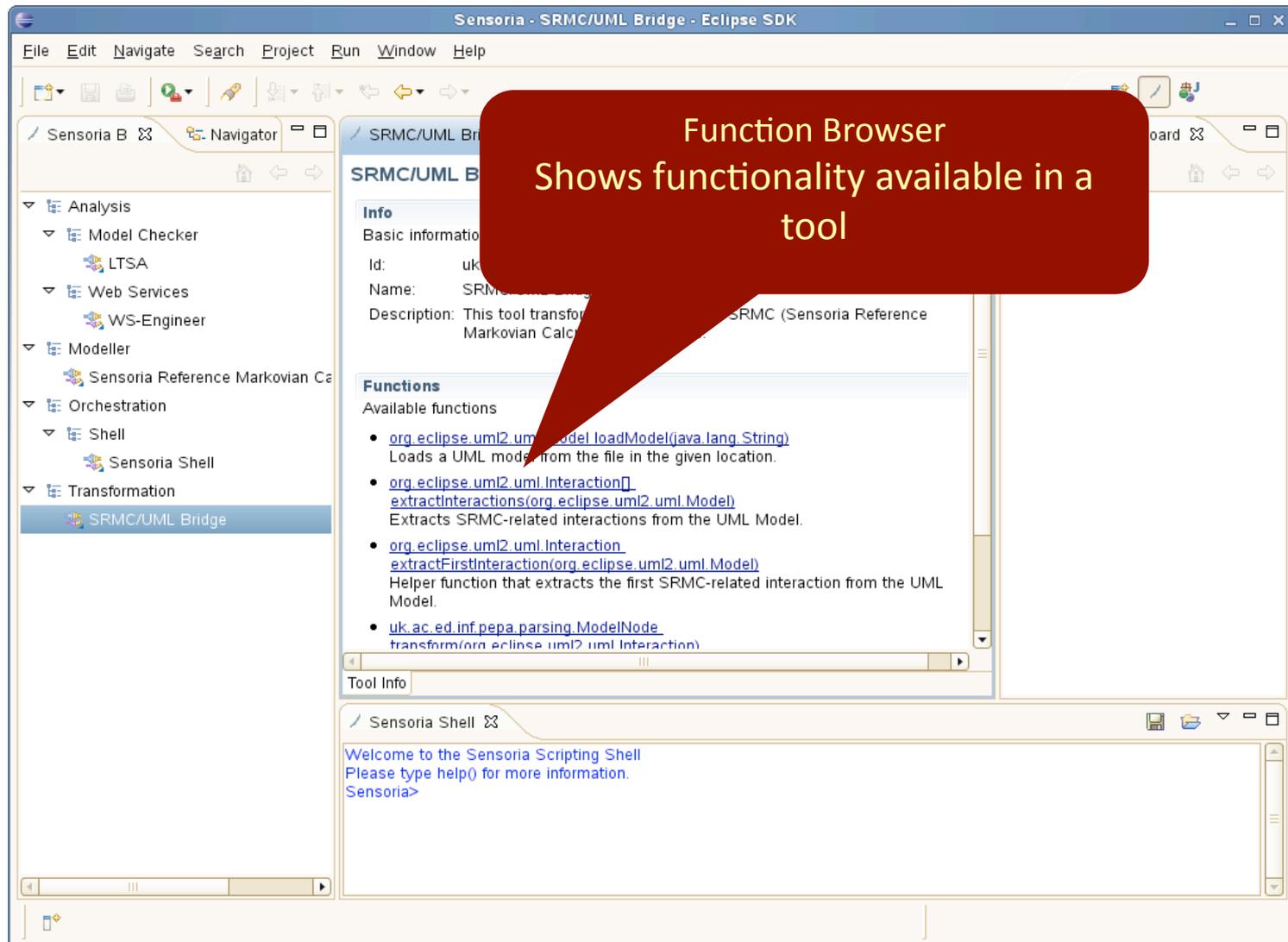
- [org.eclipse.uml2.uml.modelloader.ModelLoader\(java.lang.String\)](#)
Loads a UML model from the file in the given location.
- [org.eclipse.uml2.uml.interaction.extractInteractions\(org.eclipse.uml2.uml.Model\)](#)
Extracts SRMC-related interactions from the UML Model.
- [org.eclipse.uml2.uml.interaction.extractFirstInteraction\(org.eclipse.uml2.uml.Model\)](#)
Helper function that extracts the first SRMC-related interaction from the UML Model.
- [uk.ac.ed.inf.pepa.parsing.ModelNode.transform\(org.eclipse.uml2.uml.interaction\)](#)

Tool Info

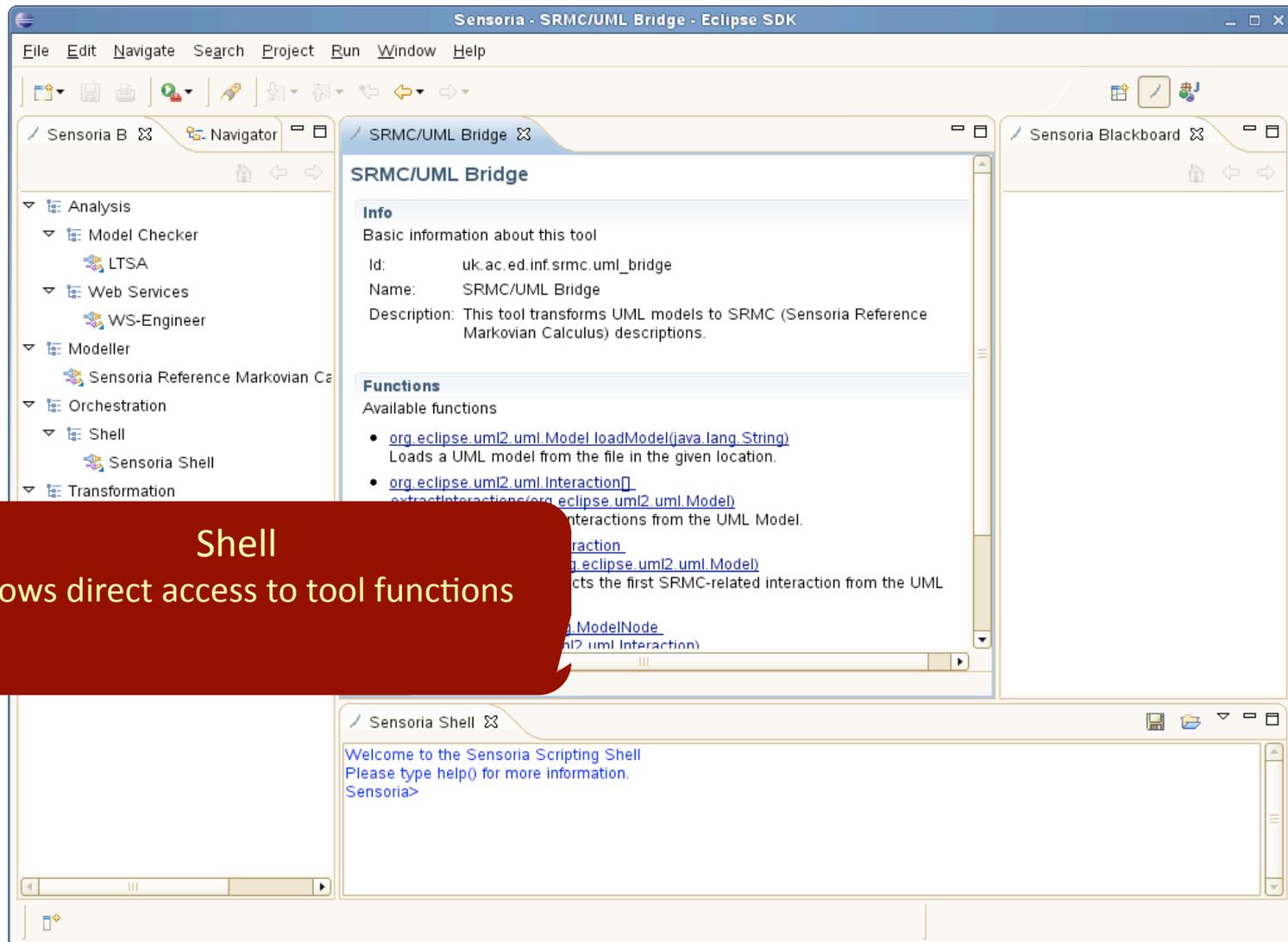
Sensoria Shell

Welcome to the Sensoria Scripting Shell
Please type help() for more information.
Sensoria>

The SDE UI



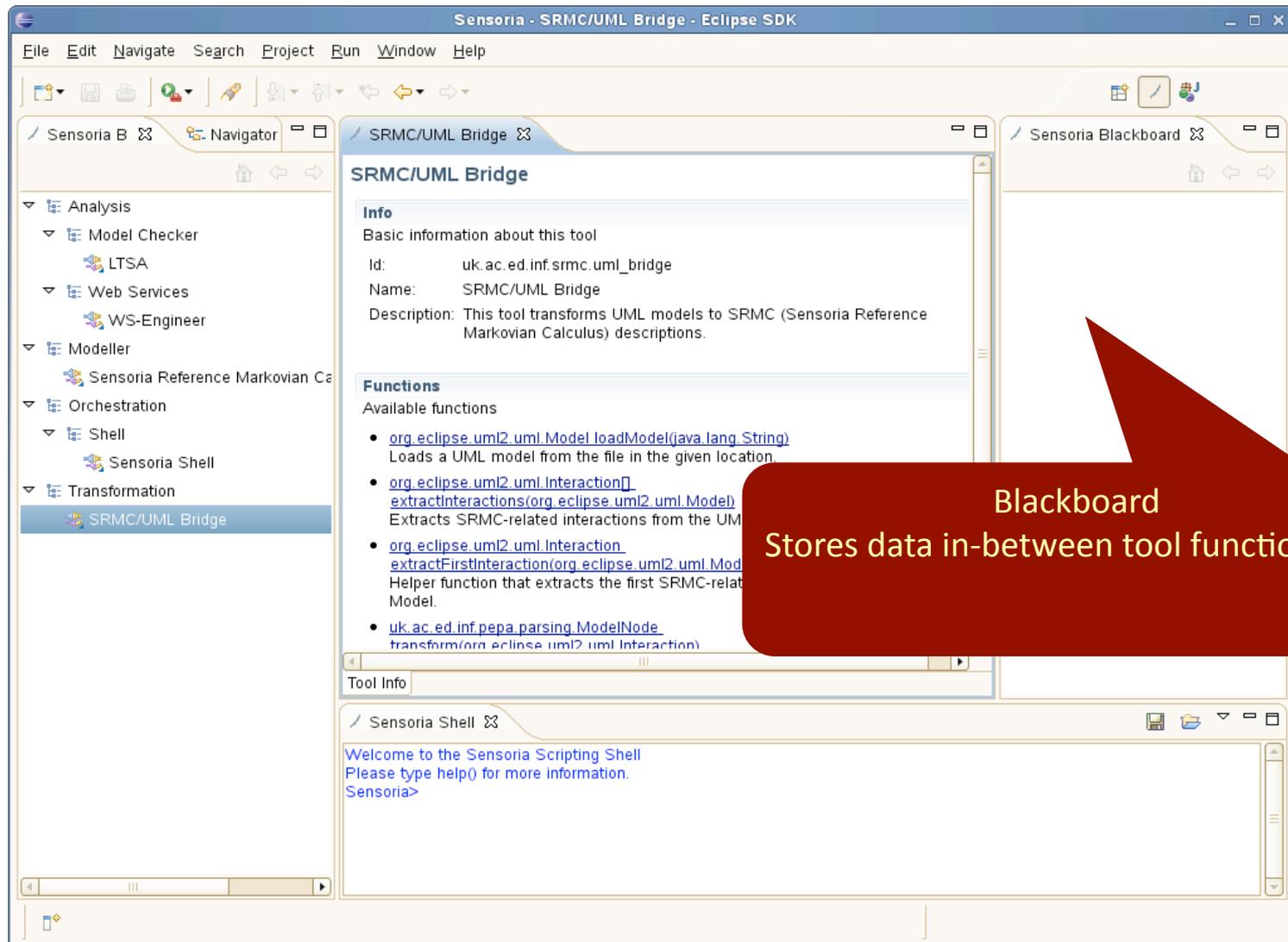
The SDE UI



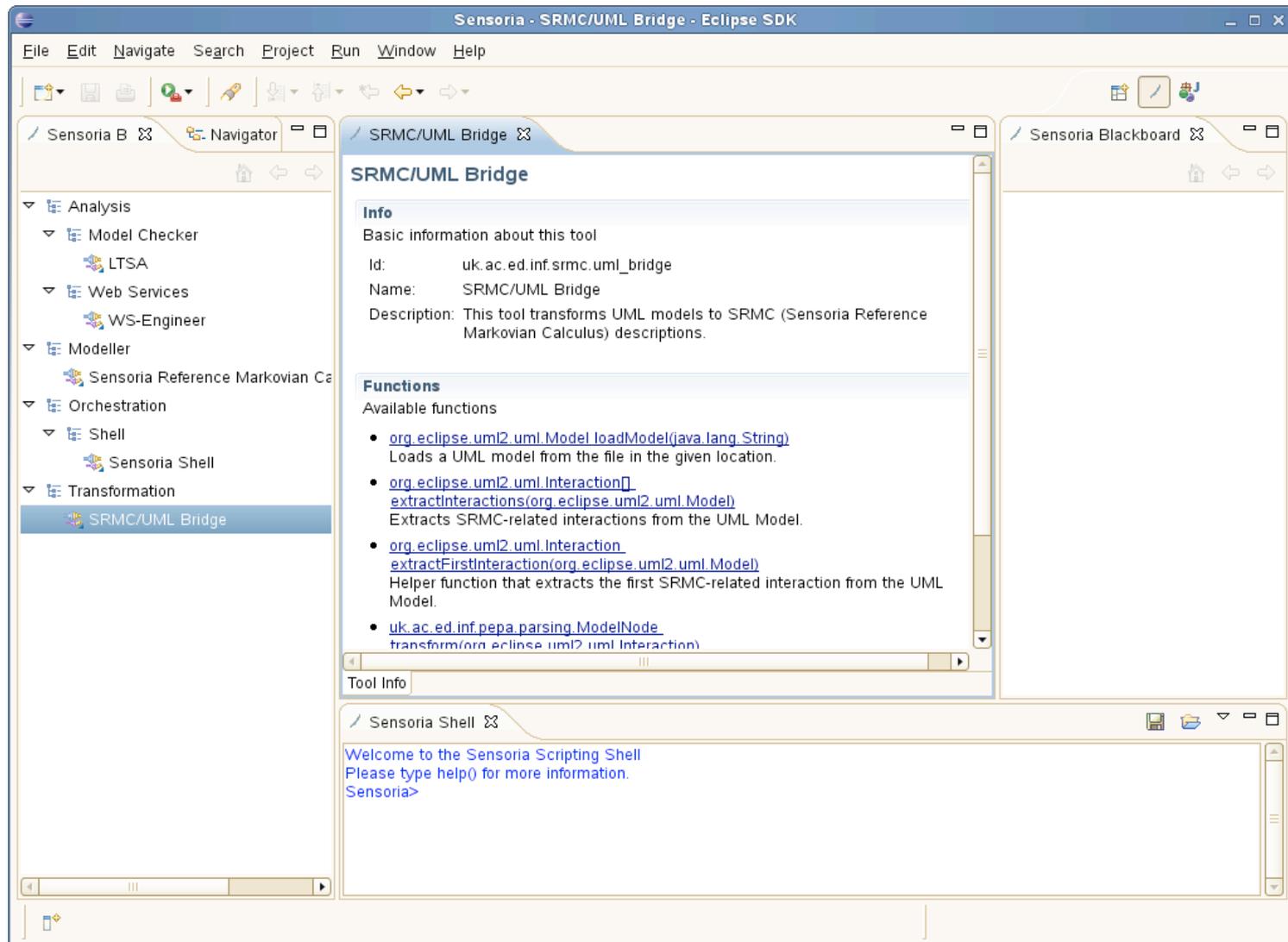
Shell

Allows direct access to tool functions

The SDE UI



The SDE UI



Correctness analysis with WS-Engineer

- **WS-Engineer** supports model-checking WS-BPEL, WS-CDL and WSIF behavioural specifications
 - Properties for analysis of service interaction & behaviour
 - Verification and violation traces
 - Animation for validation
- Automotive Case Study – On Road Assistance Scenario:
 - *Is it true that the credit card of the driver won't be charged if there are no available garages?*
 - Generate formal models of service orchestrations
 - Specify a property for verification
 - e.g. "**property**
service.orderGarage→**throw_fault**→**bank.revokeCharge**
"
 - Check model for violation traces of this property

Performance analysis with PEPA

- PEPA is a formal language for quantitative analysis of systems
 - A model is expressed in terms of components which perform timed activities and co-operate with each other
- The SCT Plug-in for PEPA supports steady-state analysis, that can answer questions such as:
 - What is the percentage of time that the local discovery server is idle in the long run? (*Utilisation Analysis*)
 - What is the throughput at which remote services are discovered? (*Throughput Analysis*)
 - What is the probability that the system does compensation upon notification of failure?

PEPA at a Glance

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) tool. The interface is divided into several panes:

- Sensoria Browser:** Shows a project structure with folders for 'general' and 'uml', and sub-folders for 'systems' containing various UML models like 'AutomotiveCS.uml', 'BM.profile.uml', etc.
- Sensoria Blackboard:** Lists loaded models and their internal class names, such as 'Sensoria Reference Markovian Calculus (SRMC)' and 'SRMC/UML Bridge'.
- Performance Evaluation:** A 'Graph View' showing a pie chart for 'Chart 1'. The chart is divided into five segments representing different system states or events: LocalServiceNotification[1] (red), LocalFailureNotification[1] (blue), FindService[1] (green), LocalWaitRequest[1] (yellow), and LocalServiceNotification[1] (orange).
- SRMC/UML Bridge Info:** Provides basic information about the tool, including its ID (uk.ac.ed.inf.srmc), Name (Sensoria Reference Markovian Calculus (SRMC)), and Description (This tool supports SRMC, a stochastic process algebra for performance evaluation).
- SRMC/UML Bridge Functions:** Lists available functions such as 'loadModel', 'extractInteractions', 'extractFirstInteraction', and 'transform', each with a brief description of its purpose.
- Code Editor:** Displays the SRMC model for 'systems.pepa', showing a list of actions with their rates (e.g., 'doServiceNotFound = 1.0;') and a section for 'Definitions' that define transitions like 'LocalWaitRequest' and 'FindService'.

PEPA at a Glance

Interface for PEPA steady-state analysis tools

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) tool. The interface is divided into several panes:

- SRMC/UML Bridge Info:** Provides basic information about the tool, including its ID (uk.ac.ed.inf.srmc), Name (SRMC/UML Bridge), and Description (This tool transforms UML models to SRMC (Sensoria Reference Markovian Calculus) descriptions).
- SRMC/UML Bridge Functions:** Lists available functions such as `loadModel`, `extractInteractions`, `extractFirstInteraction`, `Helper`, `transform`, and `compute`.
- systems.pepa Code:** Shows the configuration of the Markovian Calculus, including parameters like `doServiceNotFound = 1.0;`, `doNotifyFailure = 1.0;`, and `doReset = 1.0;`. It also includes definitions for service requests and notifications.
- systems.uml Code:** Shows the UML model structure, including components like `Orchestrator`, `Collaboration`, and `localDiscoverySM`.
- Chart 1:** A pie chart representing the steady-state analysis results. The chart is divided into five segments, each corresponding to a state or event:

State/Event	Color
LocalServiceNotification[1]	Red
LocalFailureNotification[1]	Blue
FindService[1]	Green
LocalWaitRequest[1]	Yellow
LocalServiceNotification[1]	Purple

PEPA at a Glance

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) tool. The interface is divided into several panes:

- Sensoria Browser:** Shows a project structure with folders like 'general' and 'uml', and sub-folders like 'systems' containing various UML models.
- Sensoria Reference Markovian Calculus (SRMC) Info:** Provides basic information about the tool, including its ID, name, and description. It also lists available functions such as `uk.ac.ed.inf.pepa.parsing.ModelNode.getModel()` and `double[] getSteadyStateProbabilityDistribution()`.
- SRMC/UML Bridge Info:** Provides information about the bridge tool, including its ID, name, and description. It lists available functions like `org.eclipse.uml2.uml.Model.loadModel()` and `org.eclipse.uml2.uml.Interaction.extractInteractions()`.
- Performance Evaluation:** A pie chart titled 'Chart 1' showing the distribution of system states. The legend includes: LocalServiceNotification[1], LocalFailureNotification[1], FindService[1], and LocalWaitRequest[1].
- Code Editor:** Displays the SRMC model for 'systems.pepa', showing parameters like `doServiceNotFound = 1.0;` and `doReset = 1.0;`, along with definitions for various actions.
- UML Model Editor:** Displays the UML model for 'systems.uml', showing a Machine diagram with an Orchestrator and Collaboration components.

Interface of the UML converter

PEPA at a Glance

The screenshot displays the Sensoria Eclipse SDK interface. The top-left pane shows a project browser with a tree structure under 'uml' containing various profile files. The top-middle pane shows the 'Sensoria Reference Markovian Calculus (SRMC)' tool information, including its ID, name, and description. The top-right pane shows the 'SRMC/UML Bridge' tool information, including its ID, name, and description. The bottom-left pane shows a 'Performance Evaluation' window with a pie chart titled 'Chart 1'. The bottom-right pane shows a UML model diagram for 'systems.umt' and 'systems_reflect.umt'.

Sensoria Reference Markovian Calculus (SRMC)

Info

Basic information about this tool

Id: uk.ac.ed.inf.srmc

Name: Sensoria Reference Markovian Calculus (SRMC)

Description: This tool supports SRMC, a stochastic process algebra for performance evaluation.

Functions

Available functions

- uk.ac.ed.inf.pepa.parsing.ModelNode.getModel(java.io.File)
- uk.ac.ed.inf.pepa.parsing.ModelNode.getModel(java.lang.String)
- uk.ac.ed.inf.pepa.parsing.ModelNode.getStateSpace()
- uk.ac.ed.inf.pepa.ctmc.derivation.ISta

SRMC/UML Bridge

Info

Basic information about this tool

Id: uk.ac.ed.inf.srmc_uml_bridge

Name: SRMC/UML Bridge

Description: This tool transforms UML models to SRMC (Sensoria Reference Markovian Calculus) descriptions.

Functions

Available functions

- org.eclipse.uml2.uml.Model.loadModel(java.lang.String)
- org.eclipse.uml2.uml.Interaction[]
- org.eclipse.uml2.uml.Interaction
- uk.ac.ed.inf.pepa.parsing.ModelNode
- uk.ac.ed.inf.pepa.parsing.ModelNode

Performance Evaluation

Chart 1

State	Percentage
LocalServiceNotification[1]	~15%
LocalFailureNotification[1]	~15%
FindService[1]	~15%
LocalWaitRequest[1]	~15%
LocalServiceNotification[1]	~15%
LocalFailureNotification[1]	~15%
FindService[1]	~15%
LocalWaitRequest[1]	~15%

systems.umt

Machine> Orchestrator

collaborationLine>> <Interaction> Component Collaboration

<Connector>

<Connector>

<Lifecycle> localDiscoverySM

<analysis>> <Comment> Throughput of failureNotified = 0.1176470588235294

<analysis>> <Comment> Throughput of serviceFound = 0.1176470588235294

<analysis>> <Comment> Throughput of serviceNotFound = 0.1176470588235294

<analysis>> <Comment> Throughput of serviceNotified = 0.11764705882352937

<Lifecycle> remoteDiscoverySM

<analysis>> <Comment> Throughput of servicesLocalised = 0.0588235294117647

<analysis>> <Comment> Throughput of remoteFailureNotified = 0.0588235294117647

<analysis>> <Comment> Throughput of servicesNotLocalised = 0.0588235294117647

<analysis>> <Comment> Throughput of servicesInfoDelivered = 0.0588235294117647

<Lifecycle> orchestrator

<analysis>> <Comment> Throughput of serviceLocalisationRequested = 0.117647

<analysis>> <Comment> Throughput of reset = 0.05882352941176472

<analysis>> <Comment> Throughput of compensate = 0.05882352941176472

<analysis>> <Comment> Throughput of localServiceRequested = 0.235294117647

property localDiscoverySM : LocalDiscoverySM

property remoteDiscoverySM : RemoteDiscoverySM

property orchestrator : Orchestrator

PEPA model description automatically inferred from UML model

PEPA at a Glance

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) project. The interface is divided into several panes:

- Sensoria Browser:** Shows a project tree with folders like 'general' and 'uml', and sub-folders like 'systems' containing various UML files (e.g., AutomotiveCS.uml, BM.profile.uml, etc.).
- Sensoria Blackboard:** Lists loaded models and their classes, such as 'Sensoria Reference Markovian Calculus (SRMC)' and 'SRMC/UML Bridge'.
- Sensoria Reference Markovian Calculus (SRMC) Info:** Provides details about the tool, including its ID (uk.ac.ed.inf.srmc), name, and description. It also lists available functions like `getModel`, `getMarkovChain`, and `getSteadyStateProbabilityDistribution`.
- SRMC/UML Bridge Info:** Details the bridge tool, including its ID (uk.ac.ed.inf.srmc_uml_bridge) and functions like `loadModel`, `extractInteractions`, and `extractFirstInteraction`.
- systems.pepa:** Displays the PEPA process definition with parameters like `doServiceNotFound = 1.0;` and `doNotifyFailure = 1.0;`.
- systems.uml:** Shows the UML interaction diagram for the 'Orchestrator' component, including lifelines and messages.
- Performance Evaluation:** A pie chart and a legend showing the distribution of analysis results. The legend includes: LocalServiceNotification[1], LocalFailureNotification[1], FindService[1], and LocalWaitRequest[1].

Utilisation Analysis Results

PEPA at a Glance

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) tool. The interface is divided into several panes:

- Sensoria Browser:** Shows a project structure with folders like 'general' and 'uml', and sub-folders like 'systems' containing various UML files.
- Sensoria Blackboard:** Lists the loaded SRMC/UML Bridge and its associated classes.
- Performance Evaluation:** Includes a 'Graph View' and an 'Export' button.
- Chart 1:** A pie chart showing the distribution of throughput results for different system components. The legend includes: LocalServiceNotification[1], LocalFailureNotification[1], FindService[1], and LocalWaitRequest[1].
- Sensoria Reference Markovian Calculus (SRMC) Info:** Provides details about the tool, including its ID (uk.ac.ed.inf.srmc), Name (Sensoria Reference Markovian Calculus (SRMC)), and Description (This tool supports SRMC, a stochastic process algebra for performance evaluation).
- SRMC/UML Bridge Info:** Provides details about the bridge tool, including its ID (uk.ac.ed.inf.srmc_uml_bridge), Name (SRMC/UML Bridge), and Description (This tool transforms UML models to SRMC (Sensoria Reference Markovian Calculus) descriptions).
- Functions:** Lists available functions such as `loadModel`, `extractInteractions`, `extractFirstInteraction`, `transform`, `getSteadyStateProbabilityDistribution`, and `double`.
- Code Editor:** Shows the UML model for 'systems.pepa' with various service and failure rates defined, such as `doServiceNotFound = 1.0;`, `doNotifyFailure = 1.0;`, and `doReset = 1.0;`. It also includes definitions for local and remote services.
- UML Diagram:** Shows a sequence diagram for 'systems.uml' and 'systems_reflect.uml' with lifelines for 'orchestrator' and 'localDiscoverySM'.

Throughput results reflected back into the UML model

PEPA at a Glance

The screenshot displays the Eclipse IDE with the Sensoria Reference Markovian Calculus (SRMC) tool and its integration with UML models. The interface is divided into several panes:

- Sensoria Browser:** Shows a project structure with folders for 'general' and 'uml', and sub-folders for 'systems' containing various UML models like 'AutomotiveCS.uml', 'BM.profile.uml', etc.
- Sensoria Blackboard:** Lists loaded models and their internal class names, such as 'Sensoria Reference Markovian Calculus (SRMC)' and 'SRMC/UML Bridge'.
- Performance Evaluation:** A 'Graph View' showing a pie chart for 'Chart 1'. The chart is divided into five segments representing different system states or events: LocalServiceNotification[1] (red), LocalFailureNotification[1] (blue), FindService[1] (green), LocalWaitRequest[1] (yellow), and LocalServiceNotification[1] (orange).
- SRMC Reference Markovian Calculus (SRMC) Info:** Provides basic information about the tool, including its ID (uk.ac.ed.inf.srmc), Name (Sensoria Reference Markovian Calculus (SRMC)), and Description (This tool supports SRMC, a stochastic process algebra for performance evaluation).
- SRMC/UML Bridge Info:** Provides basic information about the bridge, including its ID (uk.ac.ed.inf.srmc_uml_bridge), Name (SRMC/UML Bridge), and Description (This tool transforms UML models to SRMC (Sensoria Reference Markovian Calculus) descriptions).
- Functions:** Lists available functions for both the SRMC tool and the SRMC/UML Bridge, such as 'loadModel', 'extractInteractions', and 'transform'.
- Code Editor:** Shows the SRMC model for 'systems.pepa' with parameters like 'doServiceNotFound = 1.0;', 'doNotifyFailure = 1.0;', and 'doReset = 1.0;'. It also includes definitions for actions like 'LocalWaitRequest' and 'FindService'.
- UML Model Editor:** Shows the UML model for 'systems.uml' with a 'Machine' diagram and a 'Collaboration' diagram. The collaboration diagram shows interactions between components like 'localDiscoverySM' and 'remoteDiscoverySM'.

Service deployment transformations

- Generate WSDL descriptors from UML4SOA
- Tool: VIATRA2
 - **VI**sual **A**utomated model **TR**ansformations
 - general-purpose model transformation framework that supports the entire life-cycle for transformations
 - specification
 - design
 - execution
 - validation
 - within and between various modeling languages
 - Eclipse subproject: <http://eclipse.org/gmt>
 - Wiki: <http://wiki.eclipse.org/VIATRA2>

VIATRA2 transformations in SDE

The screenshot displays the Rational Software Architect (RSA) IDE interface. The main workspace is divided into several panes:

- UML Diagrams:** The top-left pane shows a UML Component Diagram for a "Vehicle Communication Gateway". It includes components like "RemoteDiscovery", "ExtService", "Garage", "TowTruck", "RentACar", and "ChargeCard". A red arrow points from a callout box to the "ChargeCard" component.
- UML Class Diagram:** The top-right pane shows a UML Class Diagram for "BankService" and "CreditCard". The "CreditCard" class has two tables of attributes and methods:

chargeCard	Attribute	Type
input	CreditCardNumber	Integer
output	ChargeFailed	Boolean

revokeCharge	Attribute	Type
input	AmountCharged	Integer
	ChargeNumber	Integer
	CreditCardNumber	Integer
output	Success	Boolean

- Source Code:** The bottom-right pane shows the source code for a VIATRA2 transformation rule. The rule is named "isoutputmessage" and is defined as a pattern:

```
pattern isoutputmessage(Operation) = (  
    uml2.metamodel.Parameter(Parameter);  
    uml2.metamodel.Operation(Operation);  
    uml2.metamodel.ParameterDirectionKind(X);  
    uml2.metamodel.BehavioralFeature.ownedParameter(OP, Operation, Parameter);  
    uml2.metamodel.Parameter.direction(PD, Parameter, X);  
    check(value(X) == "return");  
)  
or  
(  
    uml2.metamodel.Parameter(Parameter);  
    uml2.metamodel.Operation(Operation);  
    uml2.metamodel.ParameterDirectionKind(X);  
    uml2.metamodel.BehavioralFeature.ownedParameter(OP, Operation, Parameter);  
    uml2.metamodel.Parameter.direction(PD, Parameter, X);  
    check(value(X) == "out");  
)
```

- Project Explorer:** The bottom-left pane shows a project tree for "uml2soa.vpml", including packages like "SOA2OWL", "SOAOWL", "asm", "datatypes", "metamodel", "model", "references", "transformations", "uml2", "metamodel", and "models".
- Palette:** The top-center pane shows a palette with various UML modeling tools like Select, Zoom, Note, UML Common, Deployment, Use Case, Component, Package, Interface, Artifact, Usage, Component Realization, Interface Realization, Association, Instance, Class, Composite Str..., and Geometric Sha...

A red callout box on the left side of the image contains the text "Source models in UML".

VIATRA2 transformations in SDE

The screenshot displays the Rational Software Architect (RSA) interface. The main workspace shows a UML component diagram for a 'Vehicle Communication Gateway' with various services and interfaces. A palette on the right provides tools for diagram manipulation. Below the diagram, a table lists the properties of a 'ChargeCard' object, including 'CreditCardNumber' (Integer) and 'ChargeFailed' (Boolean). The bottom pane shows a transformation rule in VTL (Visual Transformation Language) with two patterns for 'isoutputmessage'.

Property	Type
chargeCard	ChargeCard
CreditCardNumber	Integer
ChargeFailed	Boolean
revokeCharge	revokeCharge
AmountCharged	Integer
ChargeNumber	Integer
CreditCardNumber	Integer
Success	Boolean

```
pattern isoutputmessage(Operation) = (  
  uml2.metamodel.Parameter(Parameter);  
  uml2.metamodel.Operation(Operation);  
  uml2.metamodel.ParameterDirectionKind(X);  
  uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);  
  uml2.metamodel.Parameter.direction(PD, Parameter, X);  
  check(value(X)=="return");  
)  
or  
(  
  uml2.metamodel.Parameter(Parameter);  
  uml2.metamodel.Operation(Operation);  
  uml2.metamodel.ParameterDirectionKind(X);  
  uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);  
  uml2.metamodel.Parameter.direction(PD, Parameter, X);  
  check(value(X)=="out");  
)
```

Uniform representation of models and transformations

VIATRA2 transformations in SDE

The screenshot displays the Rational Software Architect (RSA) environment. The main workspace shows a UML Component Diagram for a 'Vehicle Communication Gateway' component. This component has several provided and required interfaces: RemoteServices (RemoteDiscovery), ConnectionExtService (ExtService), OrderGarage (Garage), OrderTowTruck (TowTruck), ChargeCard (Bank), and OrderRentACar (RentACar). Below the component are two interface definitions: 'Bank' with methods 'chargeCard()' and 'revokeCharge()', and 'TowTruck' with methods 'orderTowTruck()' and 'cancelTowTruck()'. A palette on the right lists various UML elements like Select, Zoom, Note, UML Common, Deployment, Use Case, Component, Package, Interface, Artifact, Usage, Component Realization, Interface Realization, Instance, Class, Composite Structure, and Geometric Shape.

On the right, a 'Bank_WSDL.wsdl' diagram shows a 'BankService' interface with a 'ChargeCard' operation. The 'ChargeCard' operation is detailed in a table below:

Direction	Parameter Name	Type
input	CreditCardNumber	Integer
output	ChargeFailed	Boolean
input	AmountCharged	Integer
input	ChargeNumber	Integer
input	CreditCardNumber	Integer
output	Success	Boolean

The bottom-left pane shows a project tree for 'uml2soa.vpml', including packages like SOA2OWL, SOAOWL, asm, datatypes, metamodel, model, references, transformations, and uml2. The bottom-right pane displays transformation code in a 'pattern' block:

```
pattern isoutputmessage(Operation) = {
    uml2.metamodel.Parameter(Parameter);
    uml2.metamodel.Operation(Operation);
    uml2.metamodel.ParameterDirectionKind(X);
    uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    uml2.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="return");
}
or
{
    uml2.metamodel.Parameter(Parameter);
    uml2.metamodel.Operation(Operation);
    uml2.metamodel.ParameterDirectionKind(X);
    uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    uml2.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="return");
}
```

A red callout box with a white arrow points to the 'check(value(X)=="return");' line in the code, containing the text 'Model transformation code'.

The bottom status bar shows 'Problems' and 'Viatra R2 Frameworks'.

VIATRA2 transformations in SDE

The screenshot displays the Rational Software Architect (RSA) IDE interface. The main workspace shows a UML Component Diagram for a 'Vehicle Communication Gateway' component. This component has several provided interfaces: 'RemoteDiscovery', 'ExtService', 'Garage', 'TowTruck', 'Bank', and 'RentACar'. It also has several required interfaces: 'RemoteDiscovery', 'ExtService', 'Garage', 'TowTruck', 'Bank', and 'RentACar'. Below the component diagram are two interface diagrams: 'Bank' with methods 'chargeCard()' and 'revokeCharge()', and 'TowTruck' with methods 'orderTowTruck()' and 'cancelTowTruck()'. A 'Palette' on the right side of the workspace contains various UML diagram elements like Select, Zoom, Note, UML Common, Deployment, Use Case, Component, Package, Interface, Artifact, Usage, Component Realization, Interface Realization, Association, Instance, Class, Composite Str..., and Geometric Sha....

Below the main workspace, there are two smaller windows. The left one shows a project browser for 'uml2soa.vpml' with a tree structure including 'root', 'SOA2OWL', 'SOAOWL', 'asm', 'datatypes : entity', 'metamodel', 'model', 'references', 'transformations', 'uml2', 'metamodel', 'models', 'OnRoadRepair', 'Architecture {Architecture} : Model', 'OnRoadAssistance {OnRoadAssistance} : Model', 'SENSORIA-Profile {SENSORIA-Profile} : Profile', 'UMLPrimitiveTypes {UMLPrimitiveTypes} : Model', and 'vpml'. The right one shows a code editor for 'uml2soa.vtd' with the following code:

```
pattern isoutputmessage(Operation) = {
    uml2.metamodel.Parameter(Parameter);
    uml2.metamodel.Operation(Operation);
    uml2.metamodel.ParameterDirectionKind(X);
    uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    uml2.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="return");
}
or
{
    uml2.metamodel.Parameter(Parameter);
    uml2.metamodel.Operation(Operation);
    uml2.metamodel.ParameterDirectionKind(X);
    uml2.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    uml2.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="out");
}
```

A red callout box with a white border and a red arrow pointing to the 'CreditCard' class in the UML diagram contains the text: "WSDL files are created". The 'CreditCard' class has attributes 'CreditCardNumber' (Integer) and 'ChargeFailed' (Boolean), and methods 'chargeCard()' and 'revokeCharge()'.

The bottom of the IDE shows a 'Problems' window with a single entry: 'framework0 (uml2soa.vpml)'. The status bar at the bottom indicates 'Viatra R2 Frameworks'.

VIATRA2 transformations in SDE

The screenshot displays the Rational Software Architect (RSA) IDE interface, showing a UML Component Diagram and its associated code for VIATRA2 transformations.

UML Component Diagram: The diagram shows a component named "Vehicle Communication Gateway" with several provided and required interfaces. It includes components like "RemoteServices", "ConnectionExtService", "OrderGarage", "OrderTowTruck", "ChargeCard", and "Bank". Two interfaces, "Bank" and "TowTruck", are also shown with their respective methods.

UML Class Diagram: The "CreditCard" class is shown with two associations: "chargeCard" and "revokeCharge". The "chargeCard" association has an input parameter "CreditCardNumber" (Integer) and an output parameter "ChargeFailed" (Boolean). The "revokeCharge" association has an input parameter "AmountCharged" (Integer) and an output parameter "Success" (Boolean).

Code Editor: The code editor shows the implementation of the "isoutputmessage" method in the "um12soa.vtd" file. The code is as follows:

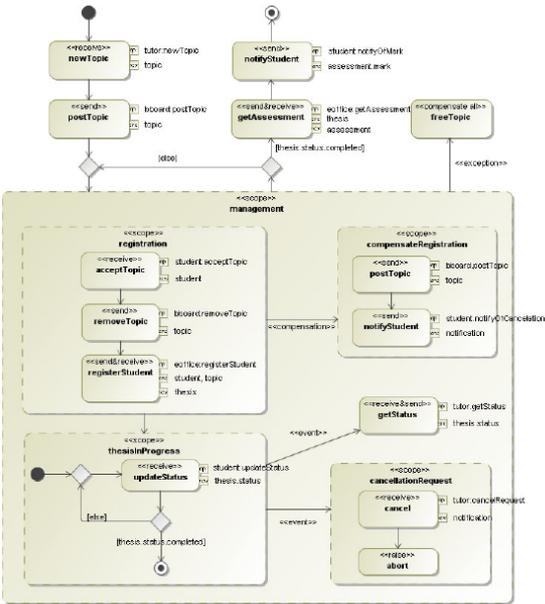
```
pattern isoutputmessage(Operation) = {
    um12.metamodel.Parameter(Parameter);
    um12.metamodel.Operation(Operation);
    um12.metamodel.ParameterDirectionKind(X);
    um12.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    um12.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="return");
}
or
{
    um12.metamodel.Parameter(Parameter);
    um12.metamodel.Operation(Operation);
    um12.metamodel.ParameterDirectionKind(X);
    um12.metamodel.BehavioralFeature.ownedParameter(OP,Operation, Parameter);
    um12.metamodel.Parameter.direction(PD, Parameter, X);
    check(value(X)=="out");
}
```

Project Explorer: The project explorer shows the structure of the "um12soa.vpml" project, including the "OnRoadRepair" package and its sub-packages like "Architecture", "OnRoadAssistance", "SENSORIA-Profile", and "UMLPrimitiveTypes".

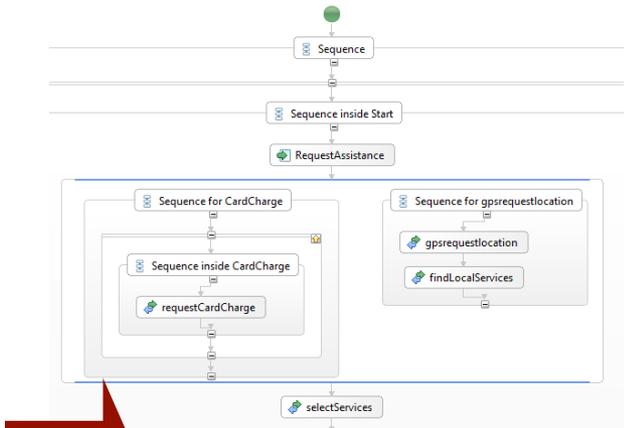
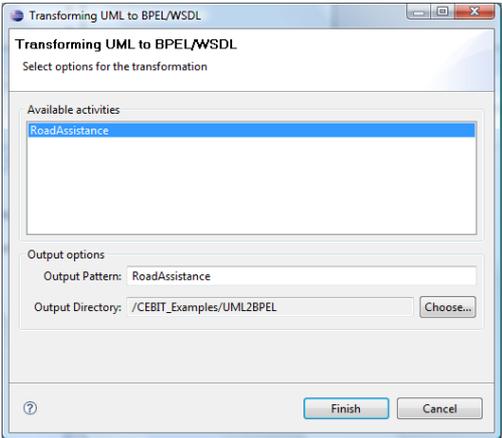
The UML2BPEL Transformation

- **The UML2BPEL transformer converts from UML activity diagrams to BPEL and WSDL**
 - Input: Service orchestrations in UML based on the UML4SOA profile
 - Output: BPEL and WSDL files
- **Integrated into Eclipse**
- **Based on Eclipse EMF, supports input models from Rational Software Architect and other tools which export EMF XMI**
- See <http://www.pst.ifi.lmu.de/projekte/uml4soa/>

UML2BPEL at a glance



Transformation



- bank_service_porttype
- requestCardCharge
- revokeCardCharge

- client_callback_porttype
- replyAssistance

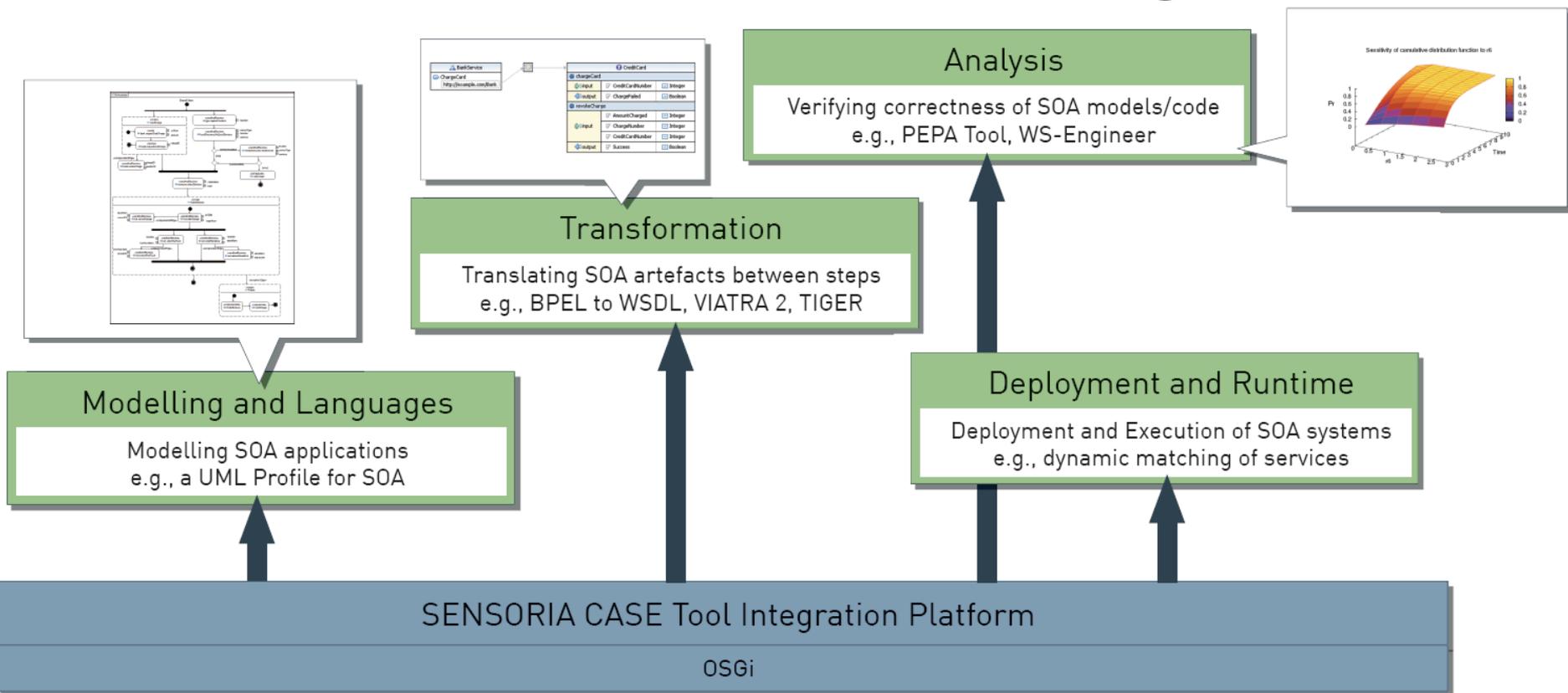
- client_service_porttype
- RequestAssistance

UML

BPEL + WSDL

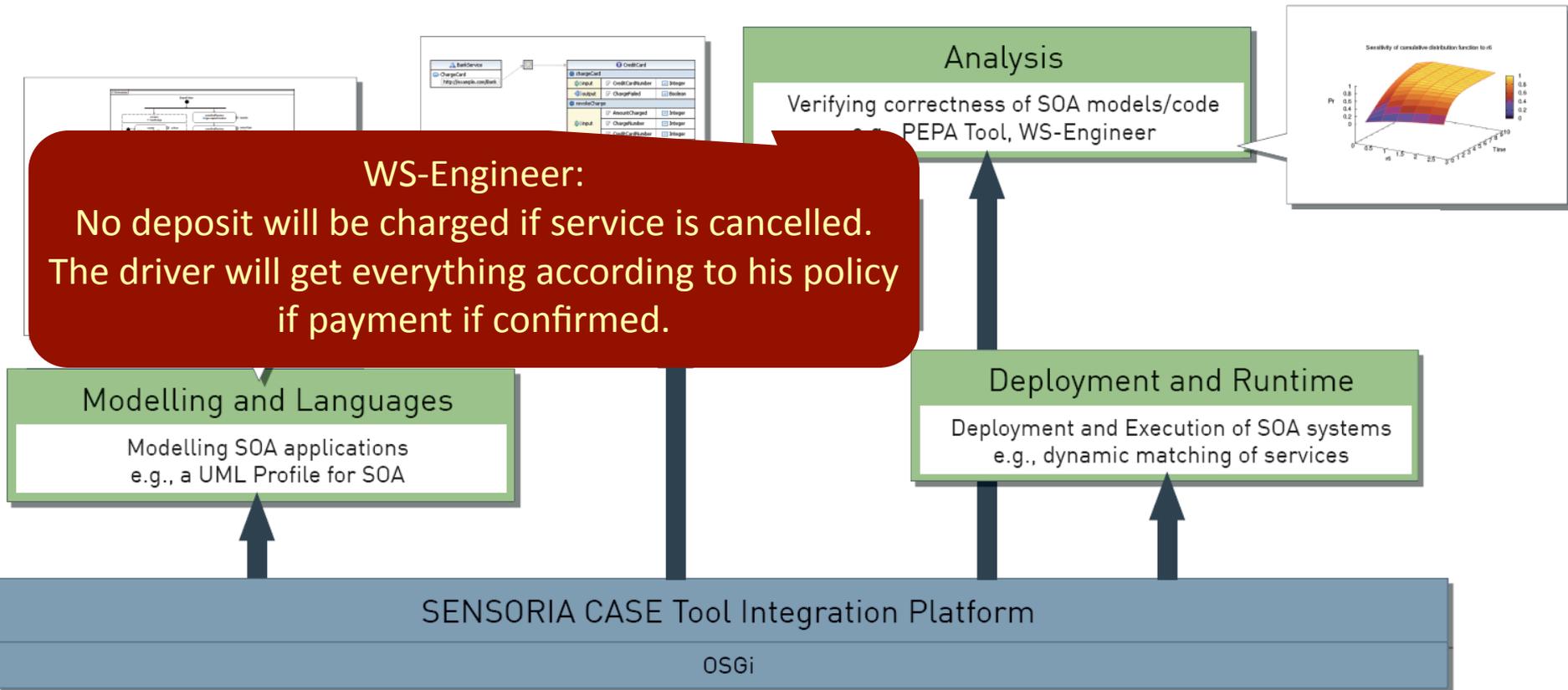
SENSORIA provides...

SENSORIA tools extend standard design workflow



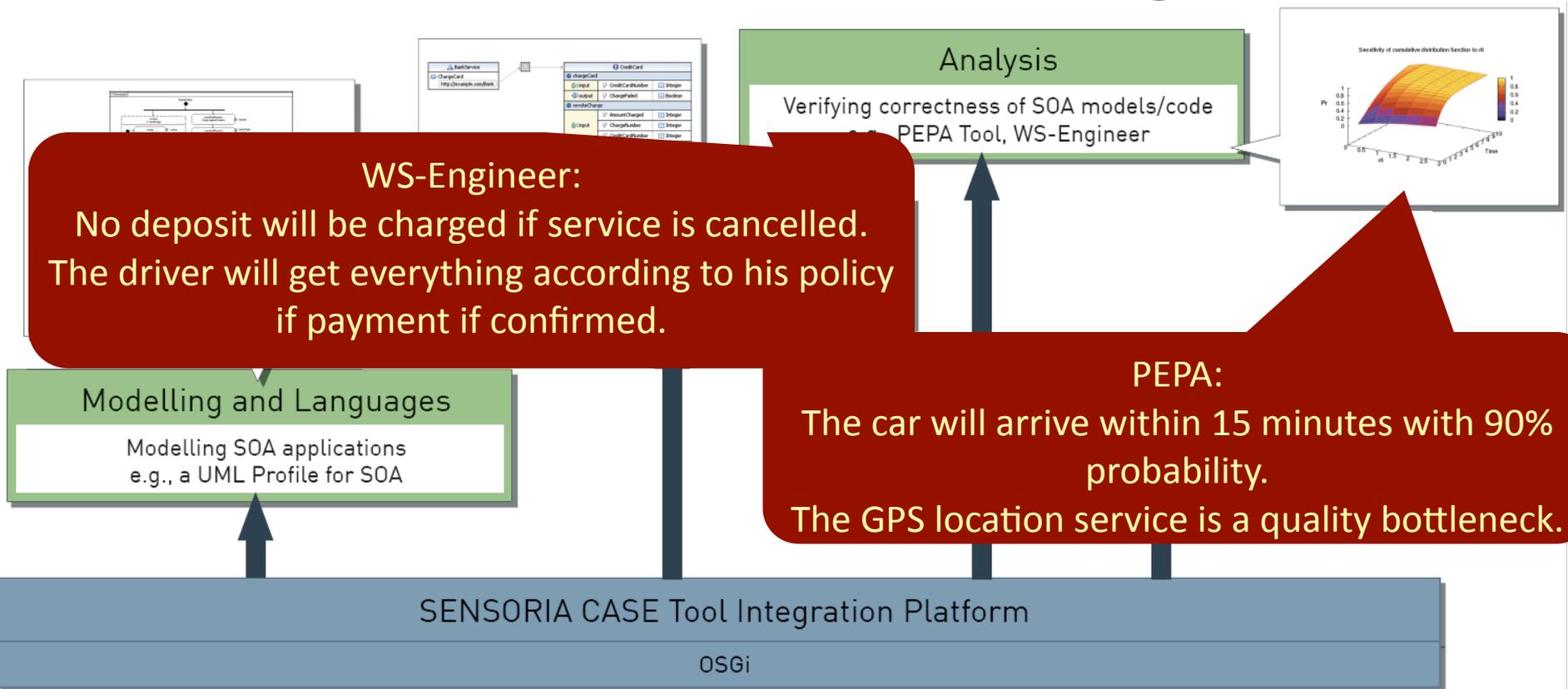
SENSORIA provides...

SENSORIA tools extend standard design workflow



SENSORIA provides...

SENSORIA tools extend standard design workflow



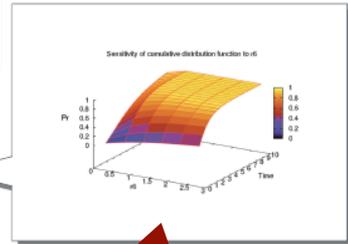
SENSORIA provides...

SENSORIA

Workflow

VIATRA:
Standard deployment code generated in WSDL.

Analysis:
Verifying correctness of SOA models/code
PEPA Tool, WS-Engine



WS-Engine:
No deposit will be charged if service is cancelled.
The driver will get everything according to his policy
if payment is confirmed.

Modelling and Languages

Modelling SOA applications
e.g., a UML Profile for SOA

PEPA:
The car will arrive within 15 minutes with 90% probability.
The GPS location service is a quality bottleneck.

SENSORIA CASE Tool Integration Platform

OSGi

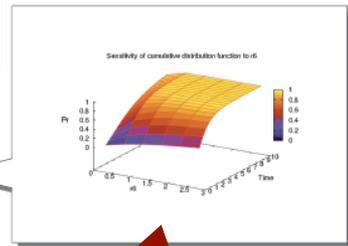
SENSORIA provides...

SENSORIA

Workflow

VIATRA:
Standard deployment code generated in WSDL.

Analysis:
Verifying correctness of SOA models/code
PEPA Tool, WS-Engine



WS-Engine:
No deposit will be charged if service is cancelled.
The driver will get everything according to his policy
if payment is confirmed.

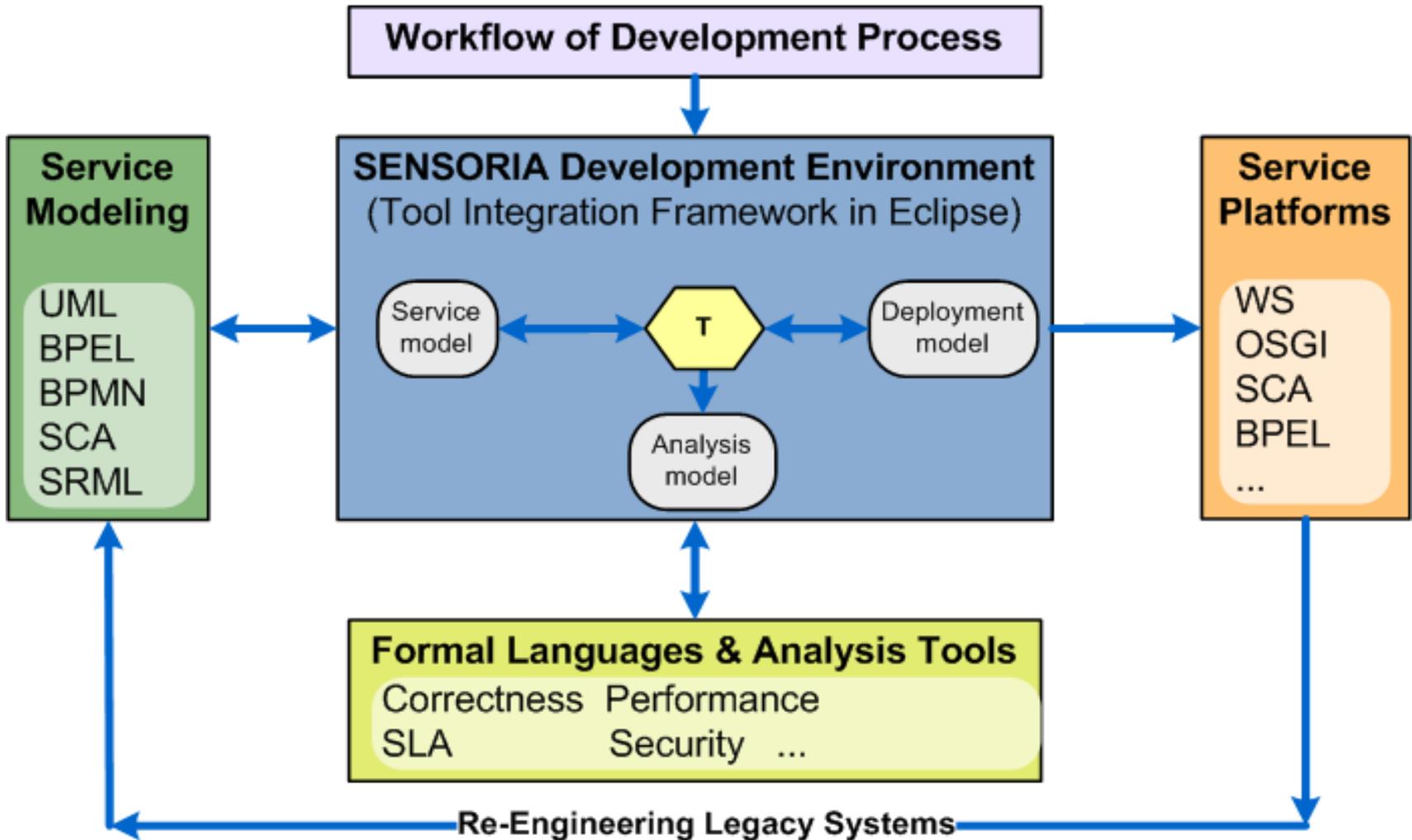
Modelling and Languages

Modelling SOA applications
e.g., a UML Profile for SOA

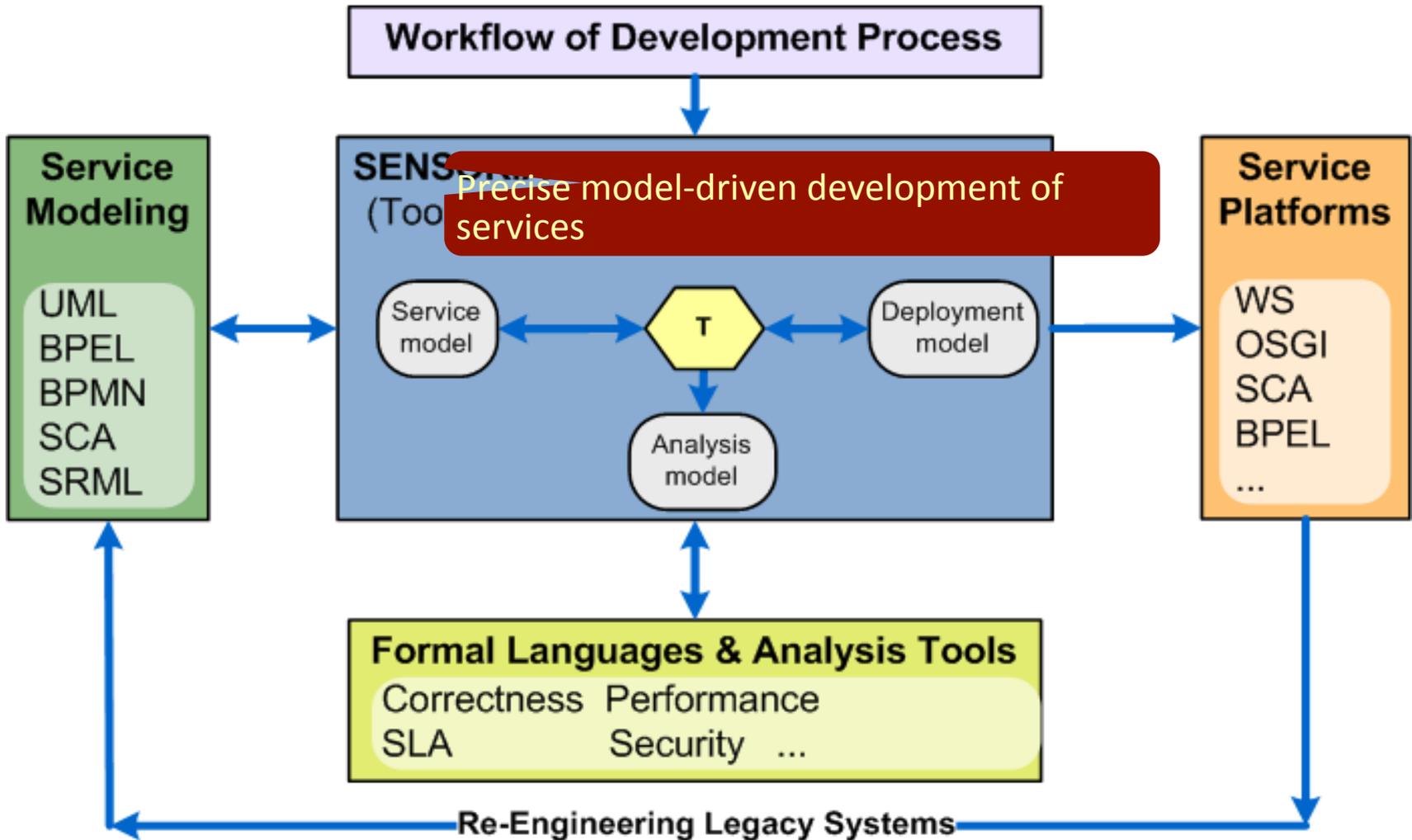
PEPA:
The car will arrive within 15 minutes with 90% probability.
The GPS location service is a quality bottleneck.

SENSORIA Development Environment:
Additional tools can easily be integrated

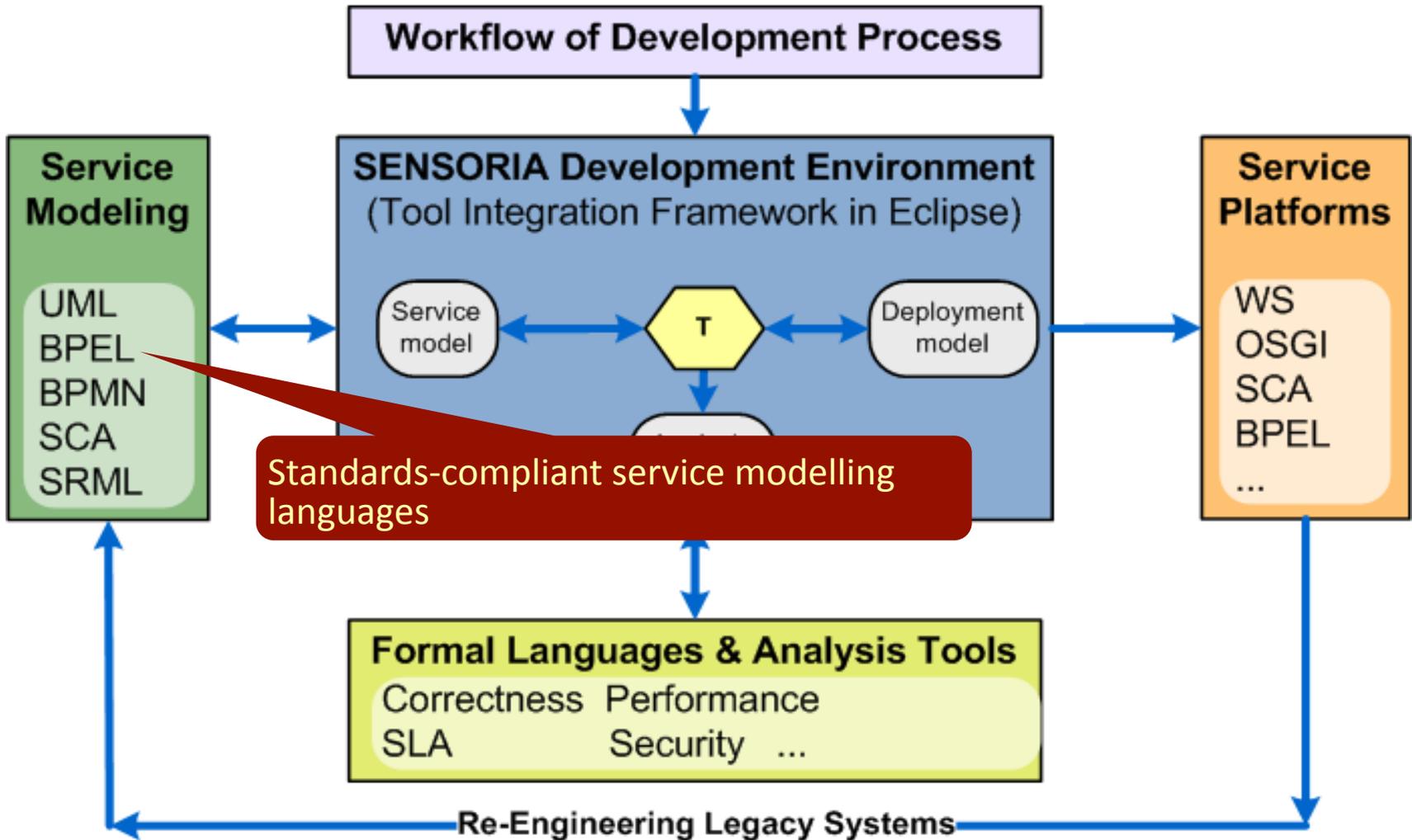
SENSORIA answers to SOA challenges



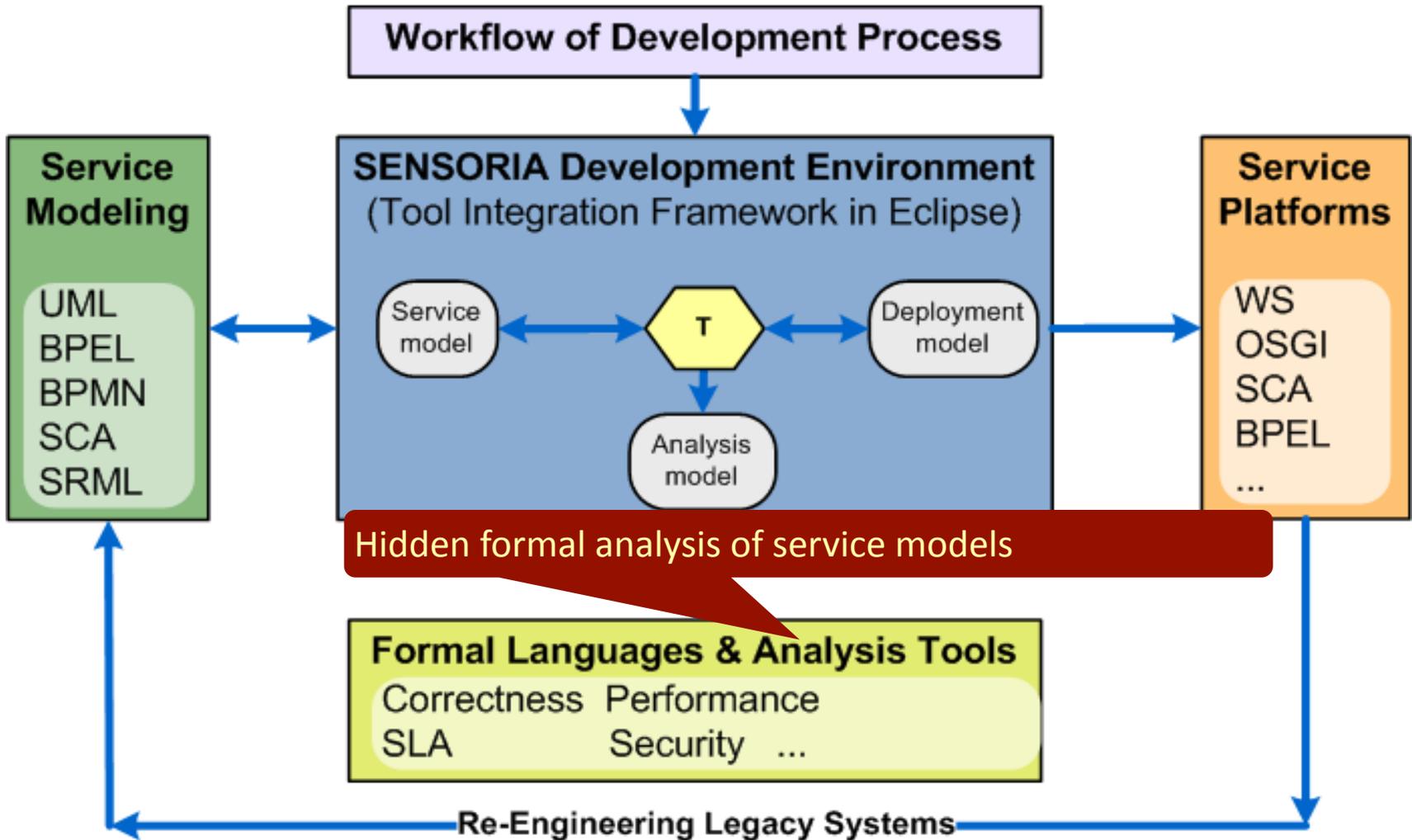
SENSORIA answers to SOA challenges



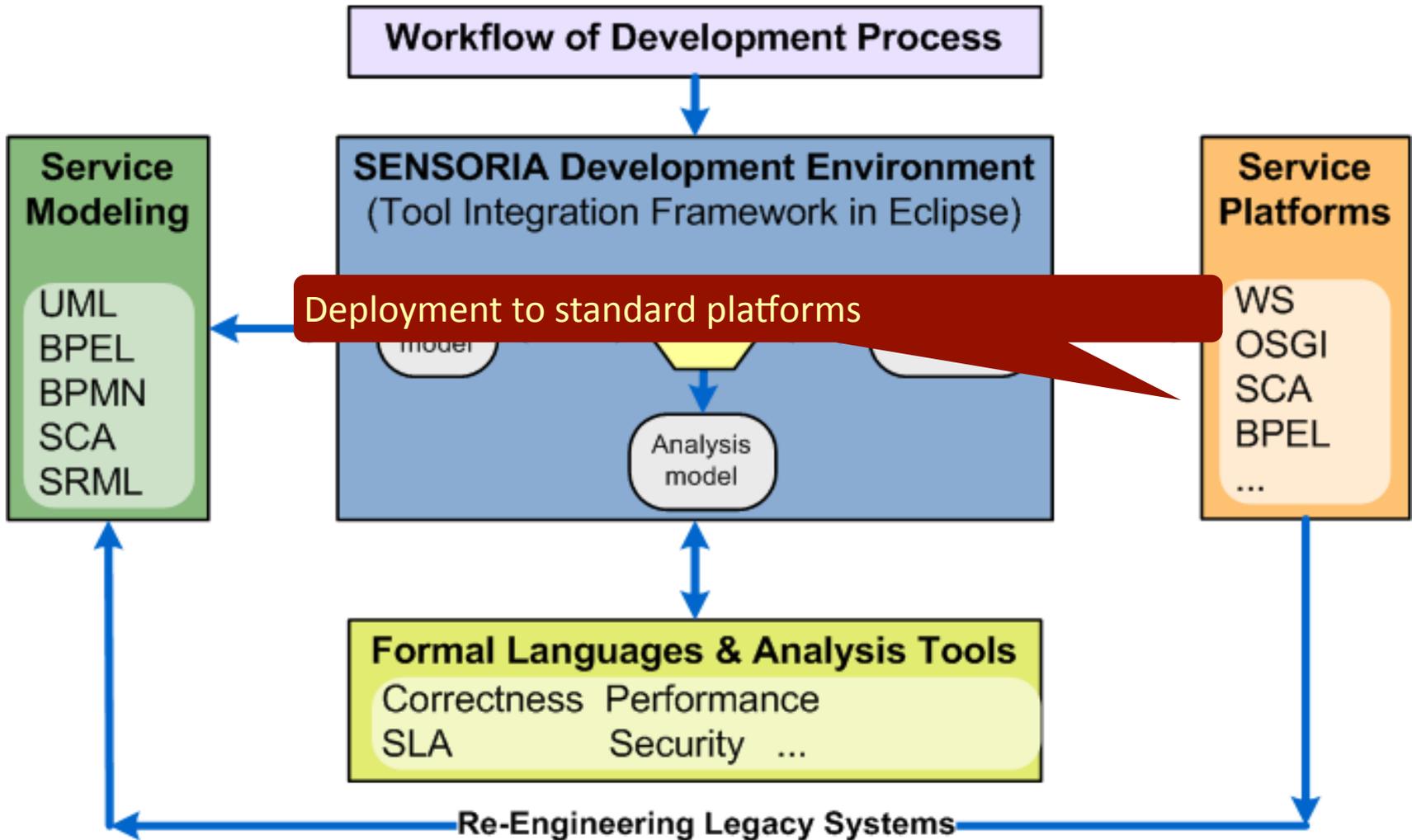
SENSORIA answers to SOA challenges



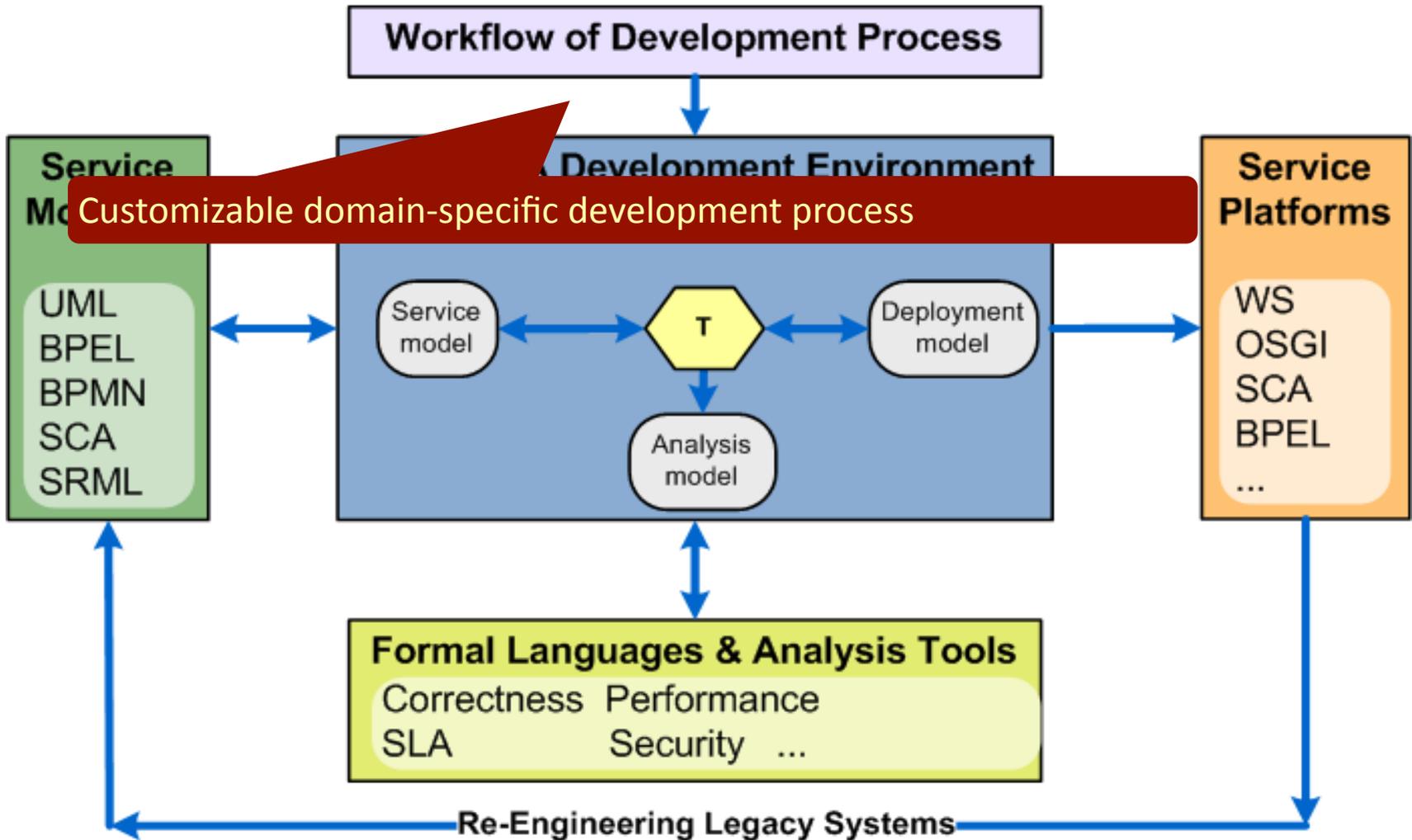
SENSORIA answers to SOA challenges



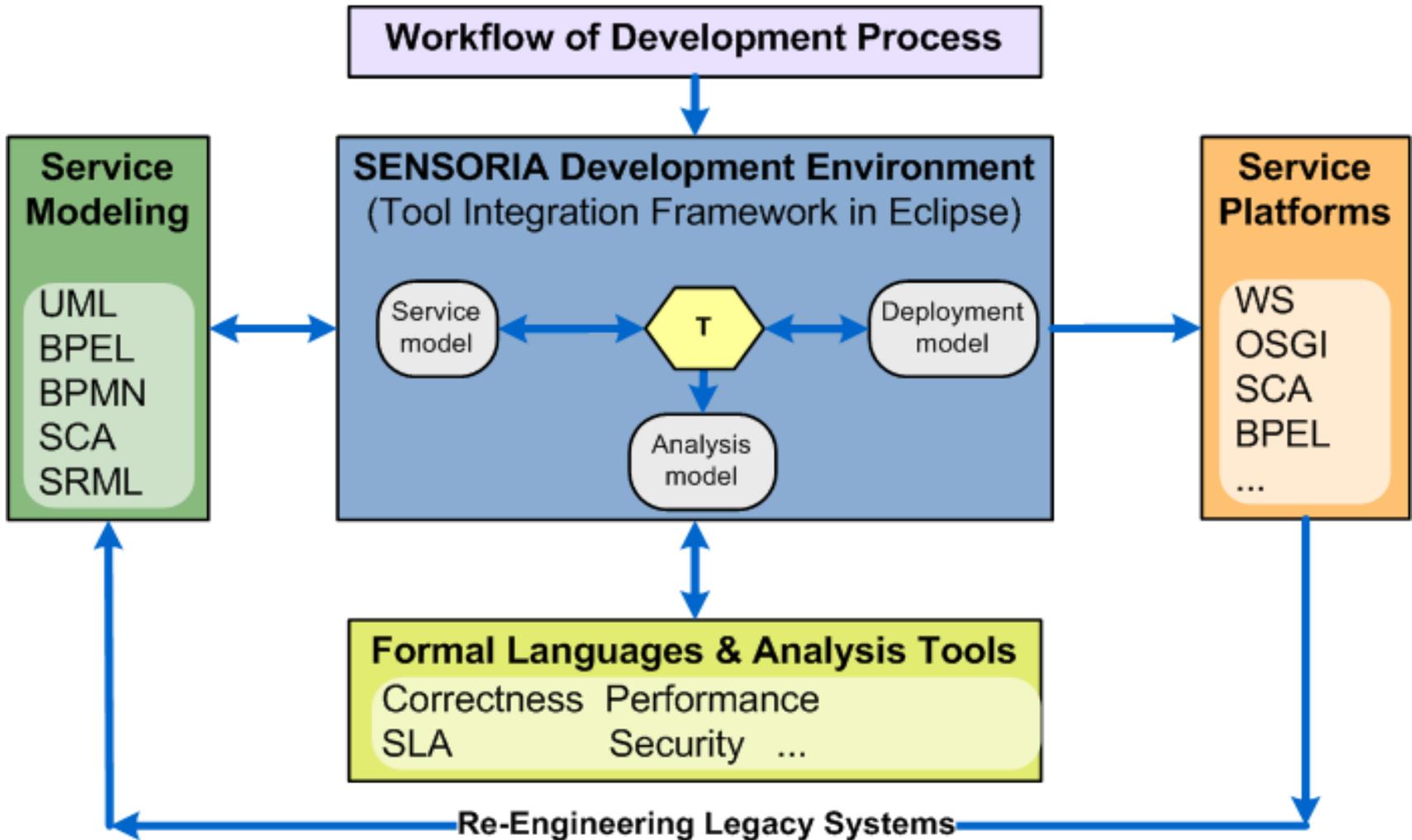
SENSORIA answers to SOA challenges



SENSORIA answers to SOA challenges



SENSORIA answers to SOA challenges

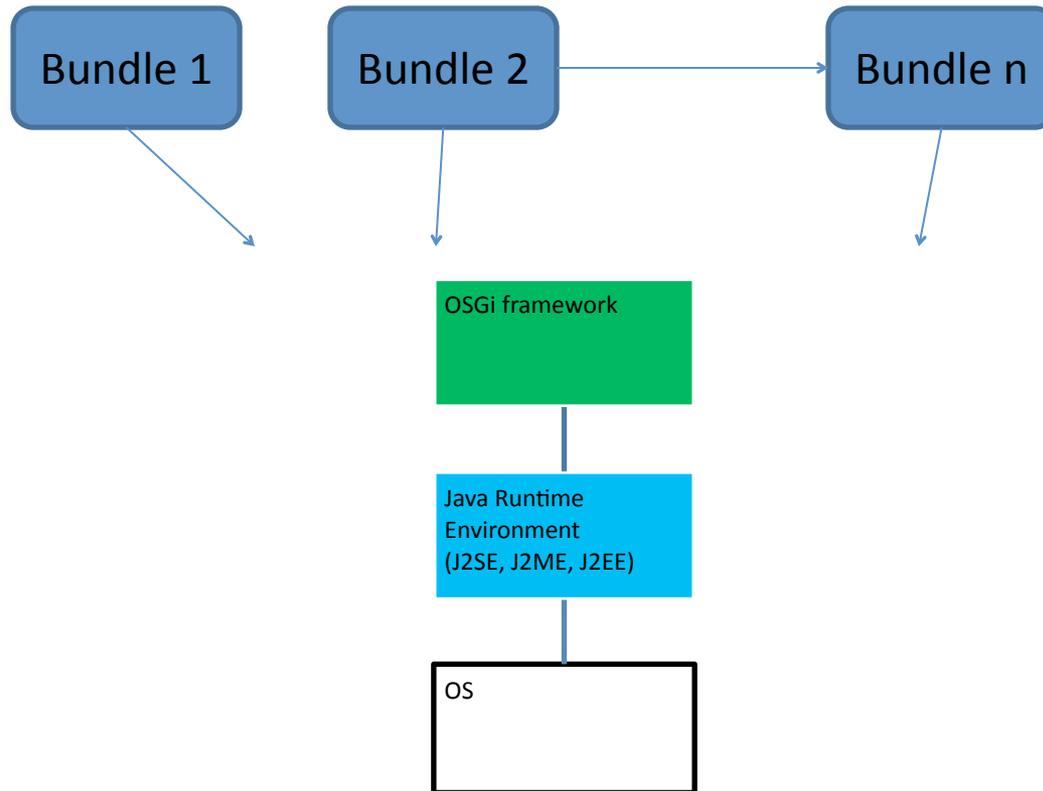


The SENSORIA Development Environment

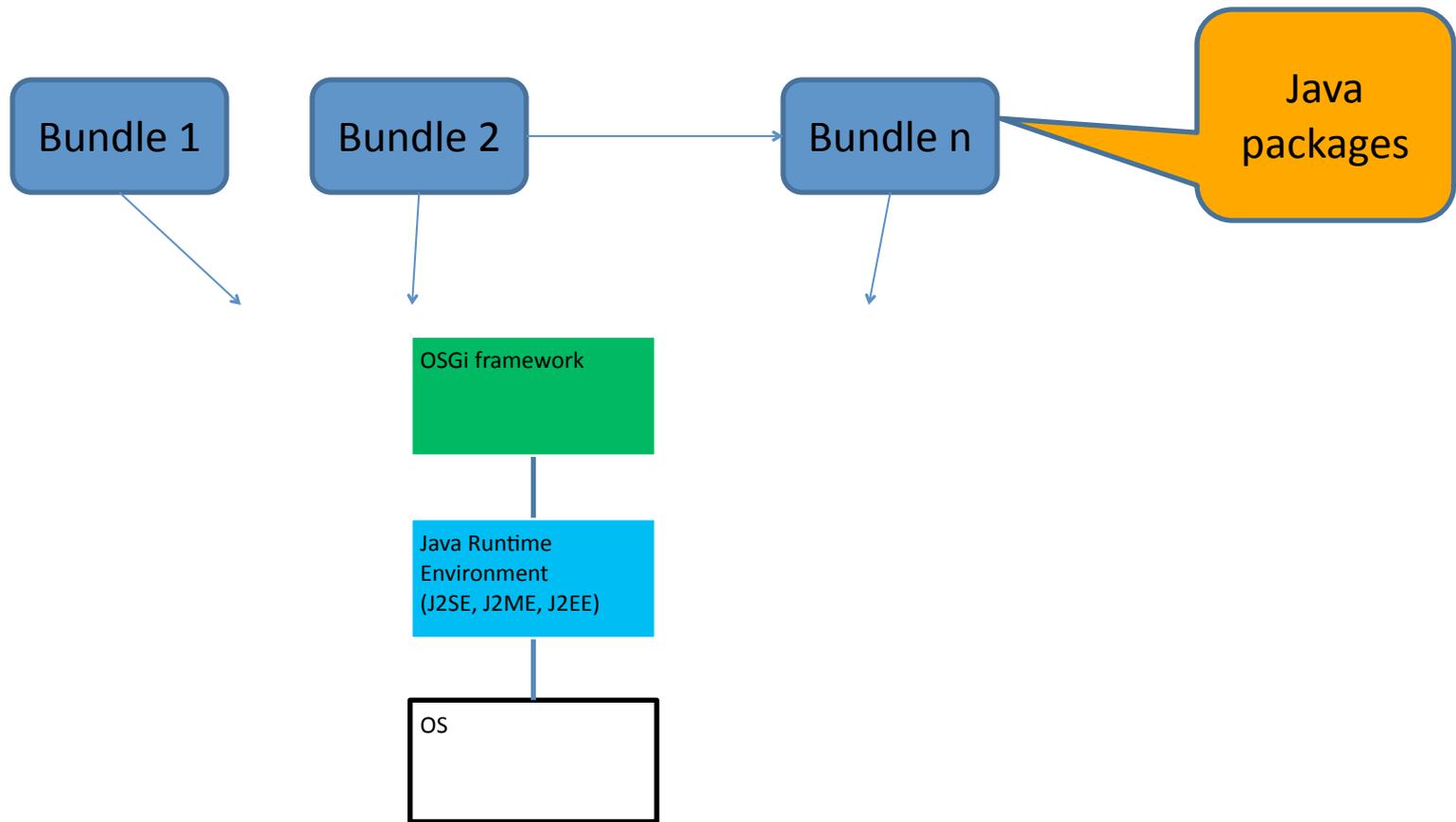
A PROGRAMMER'S PERSPECTIVE

Eclipse, OSGi as an integration platform

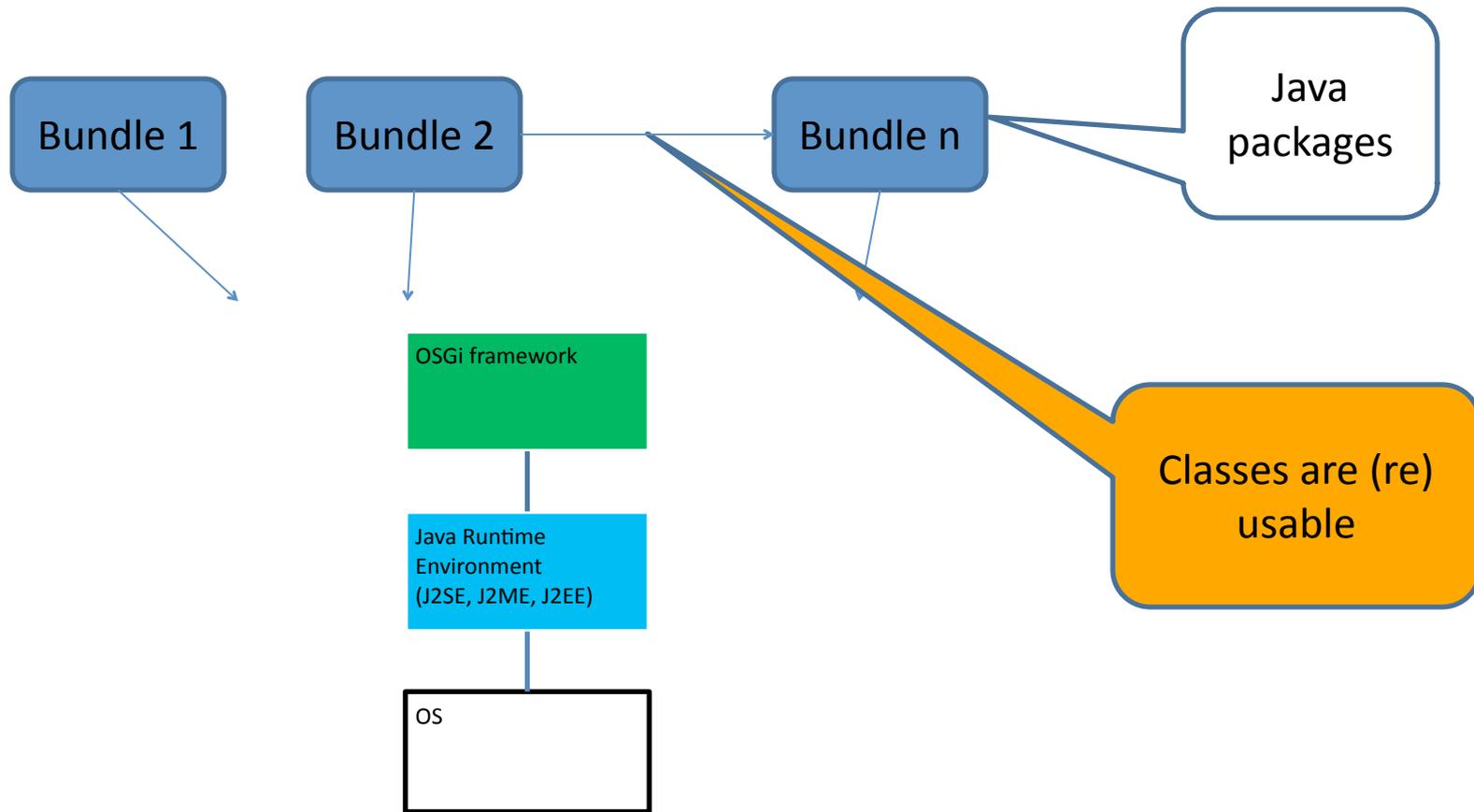
- OSGi = Open Services Gateway Initiative



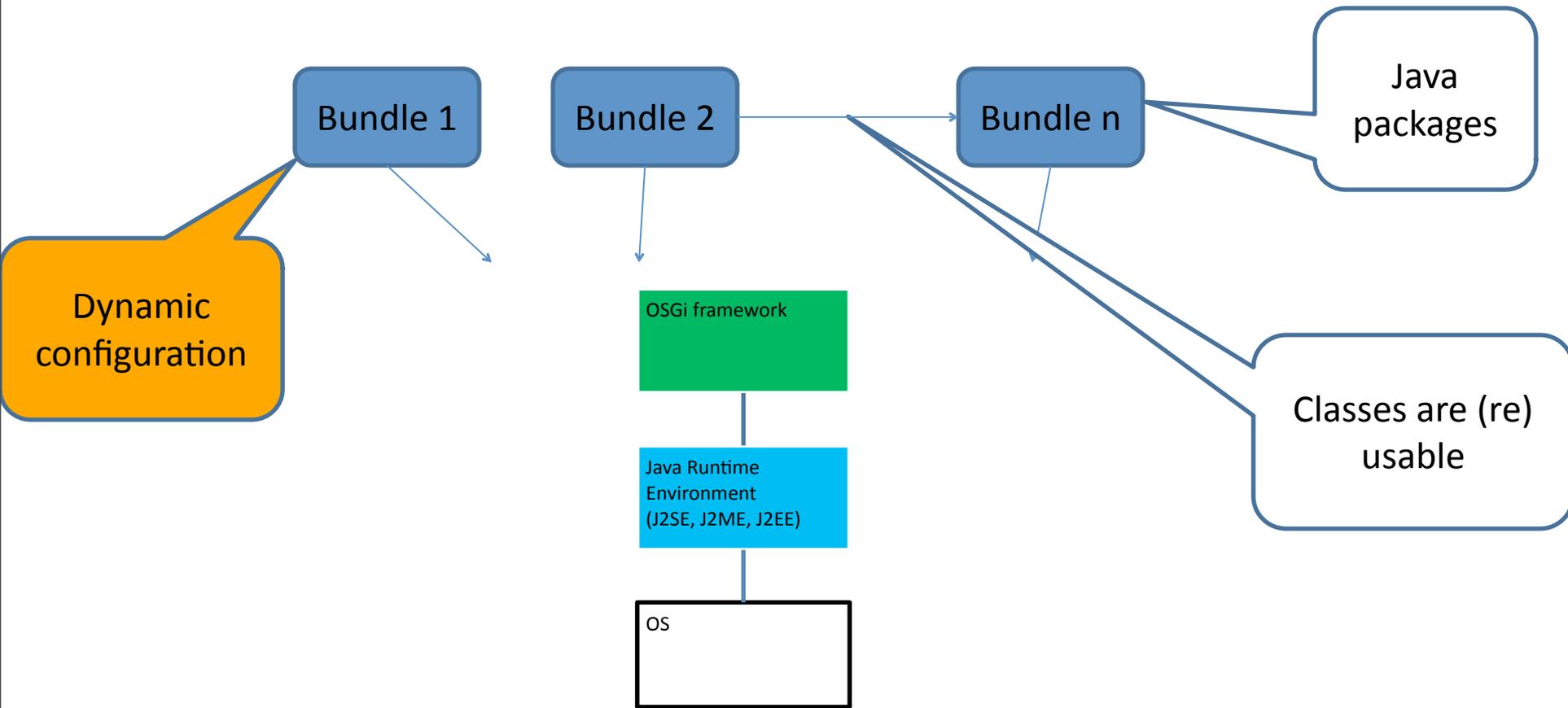
Eclipse, OSGi as an integration platform



Eclipse, OSGi as an integration platform

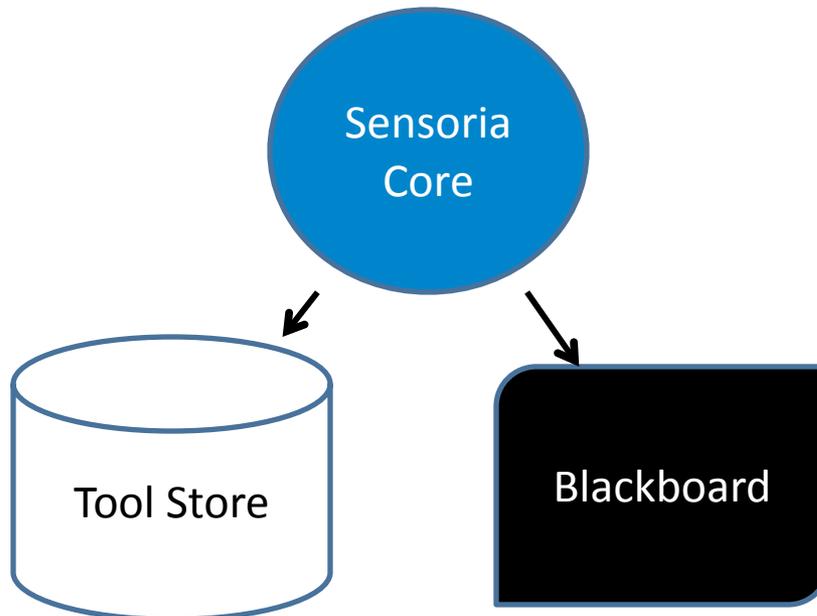


Eclipse, OSGi as an integration platform

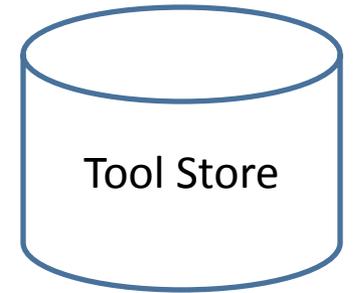


Sensoria

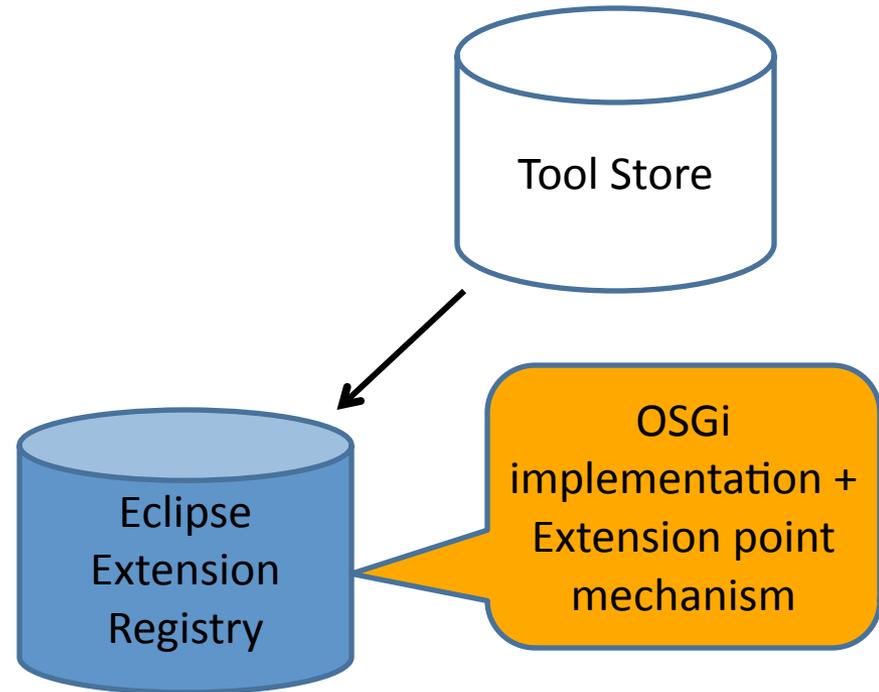
- OSGi bundles, and an Eclipse UI
- Provides basic infrastructure for OSGi-based tool and service integration



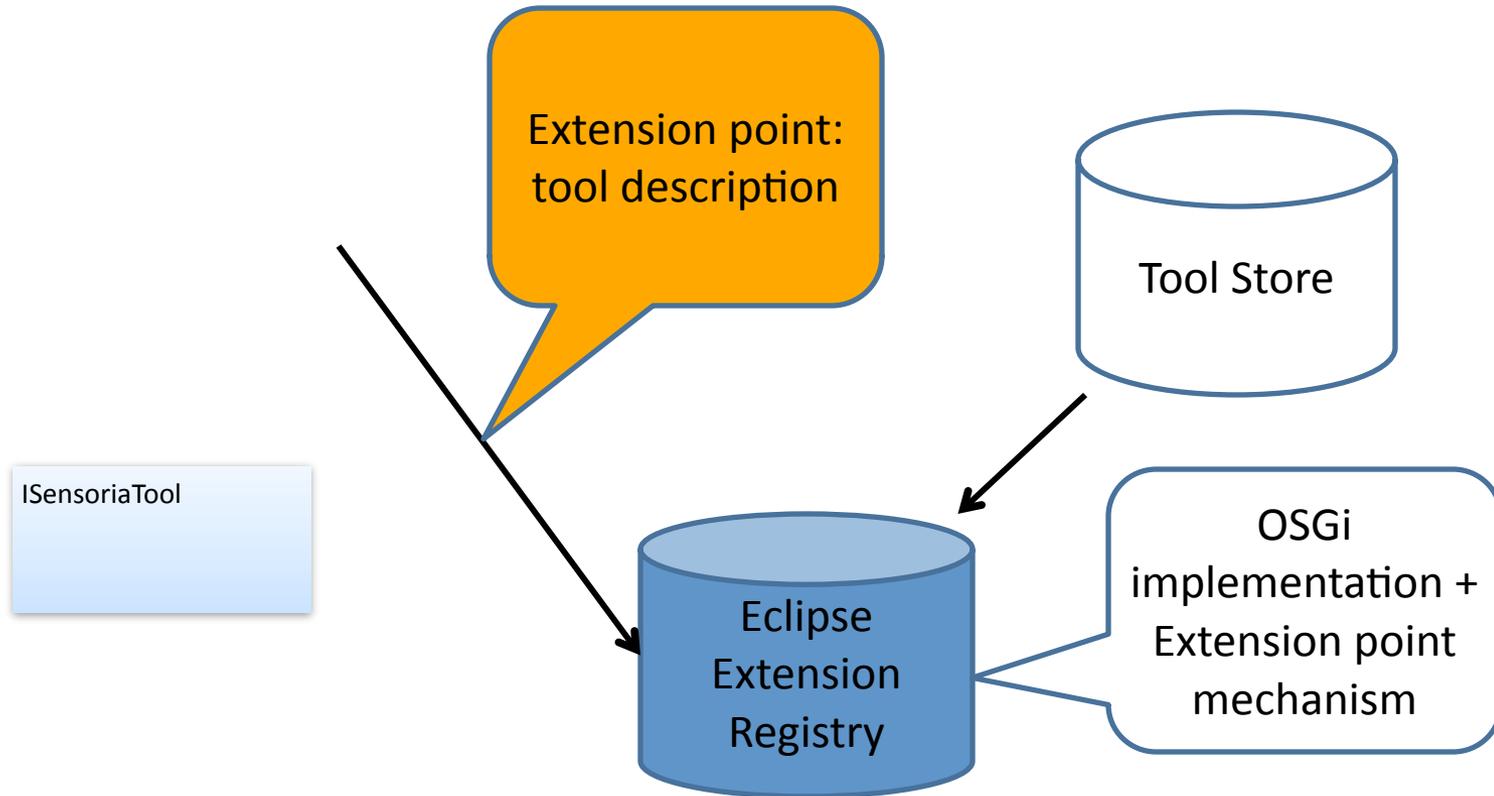
Sensoria



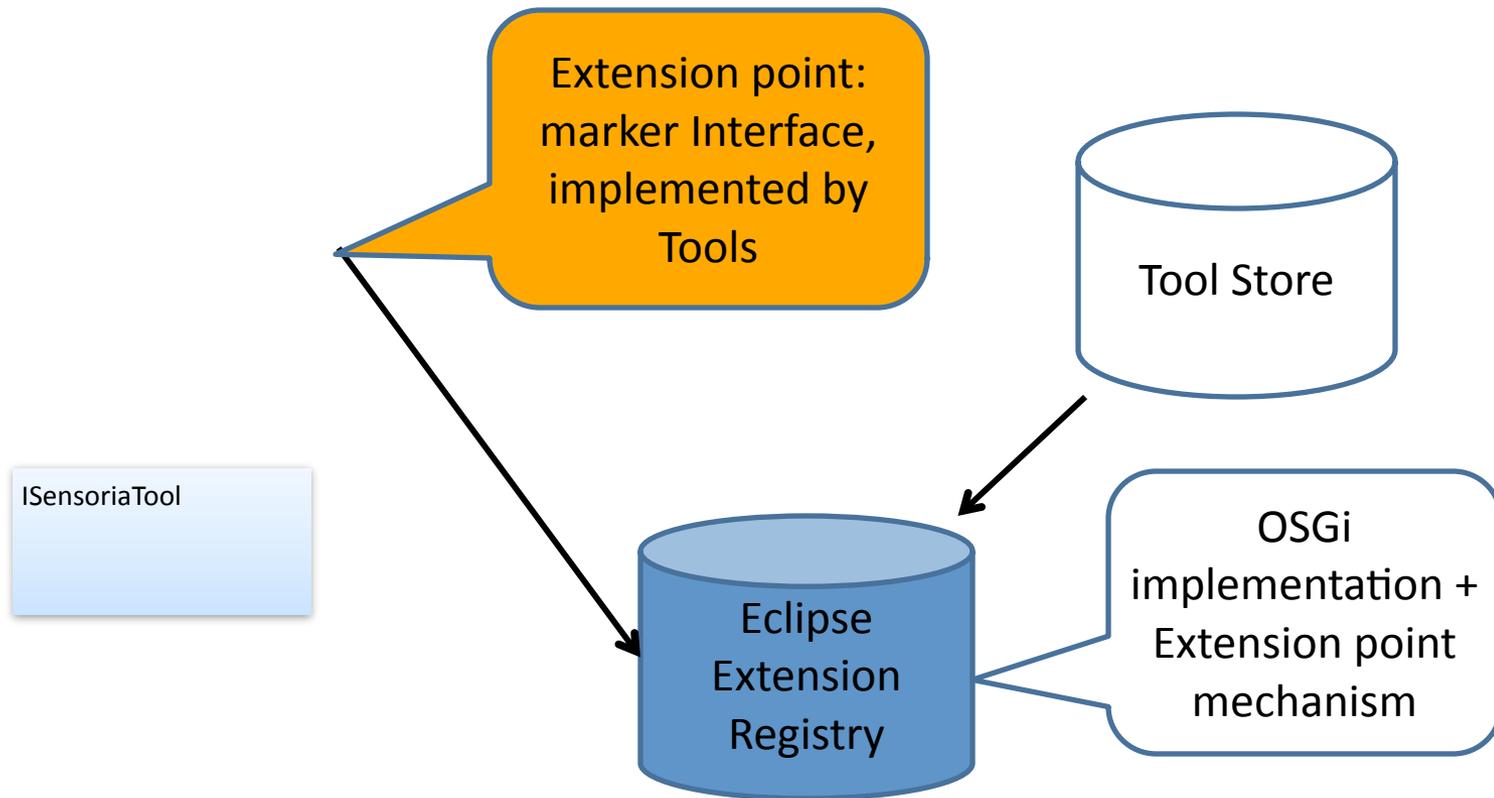
Sensoria



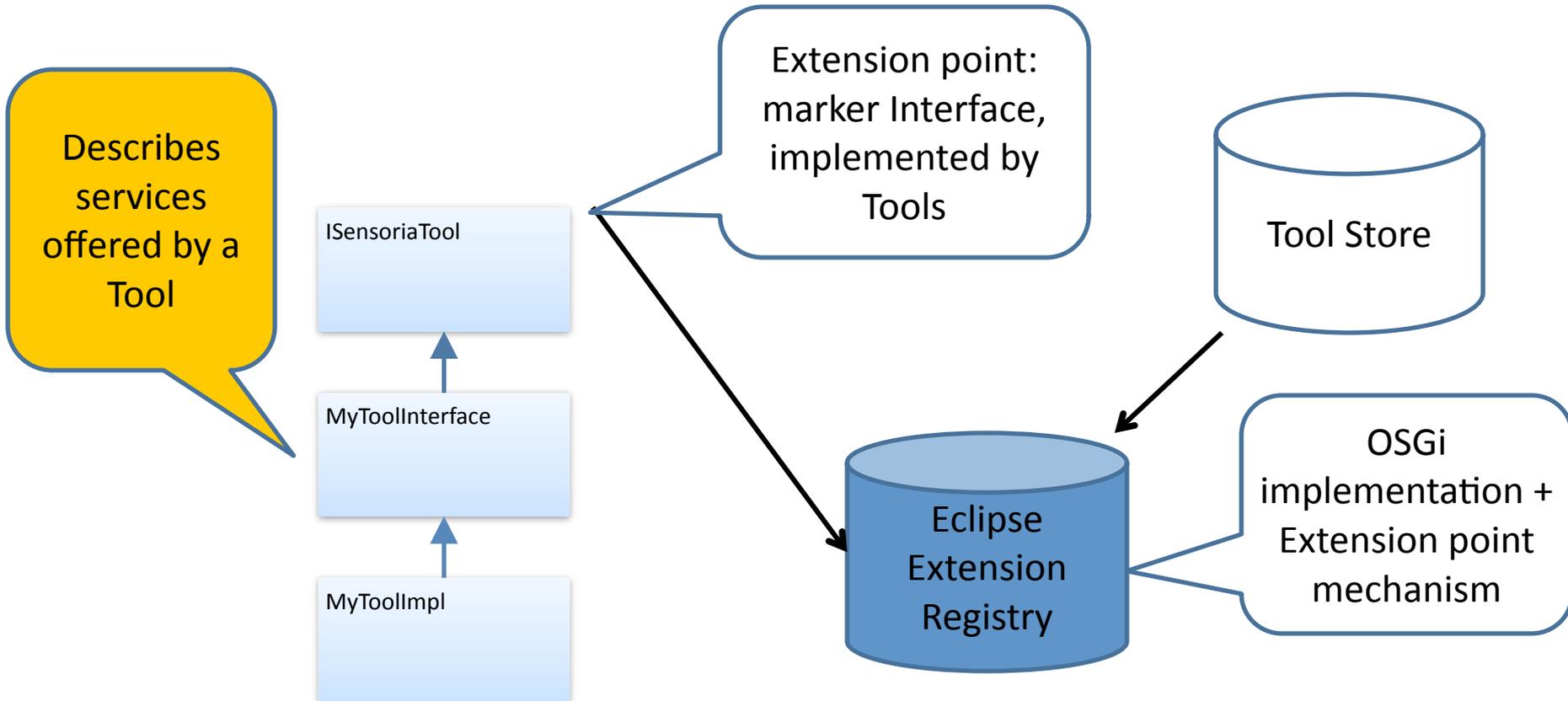
Sensoria



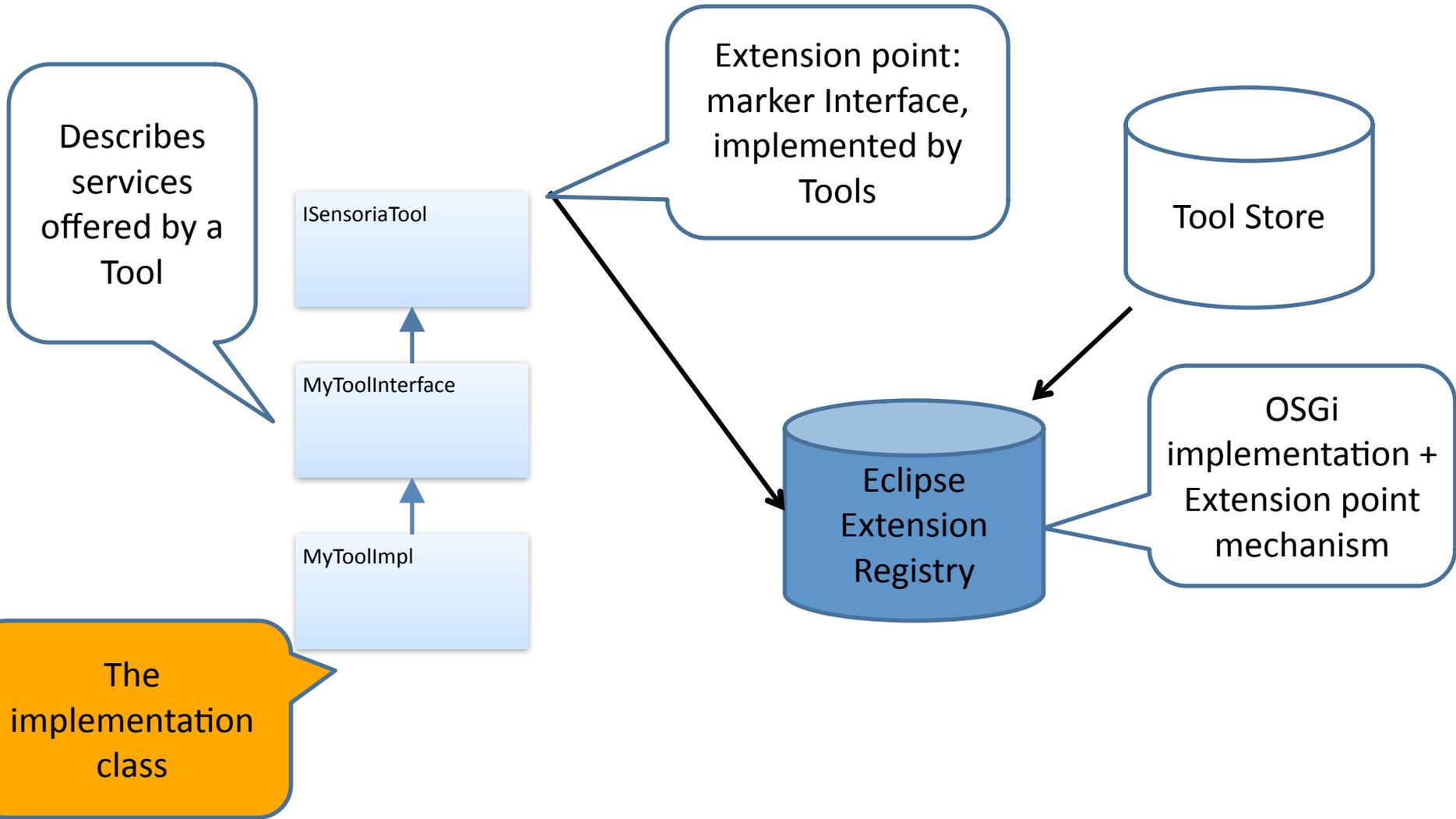
Sensoria



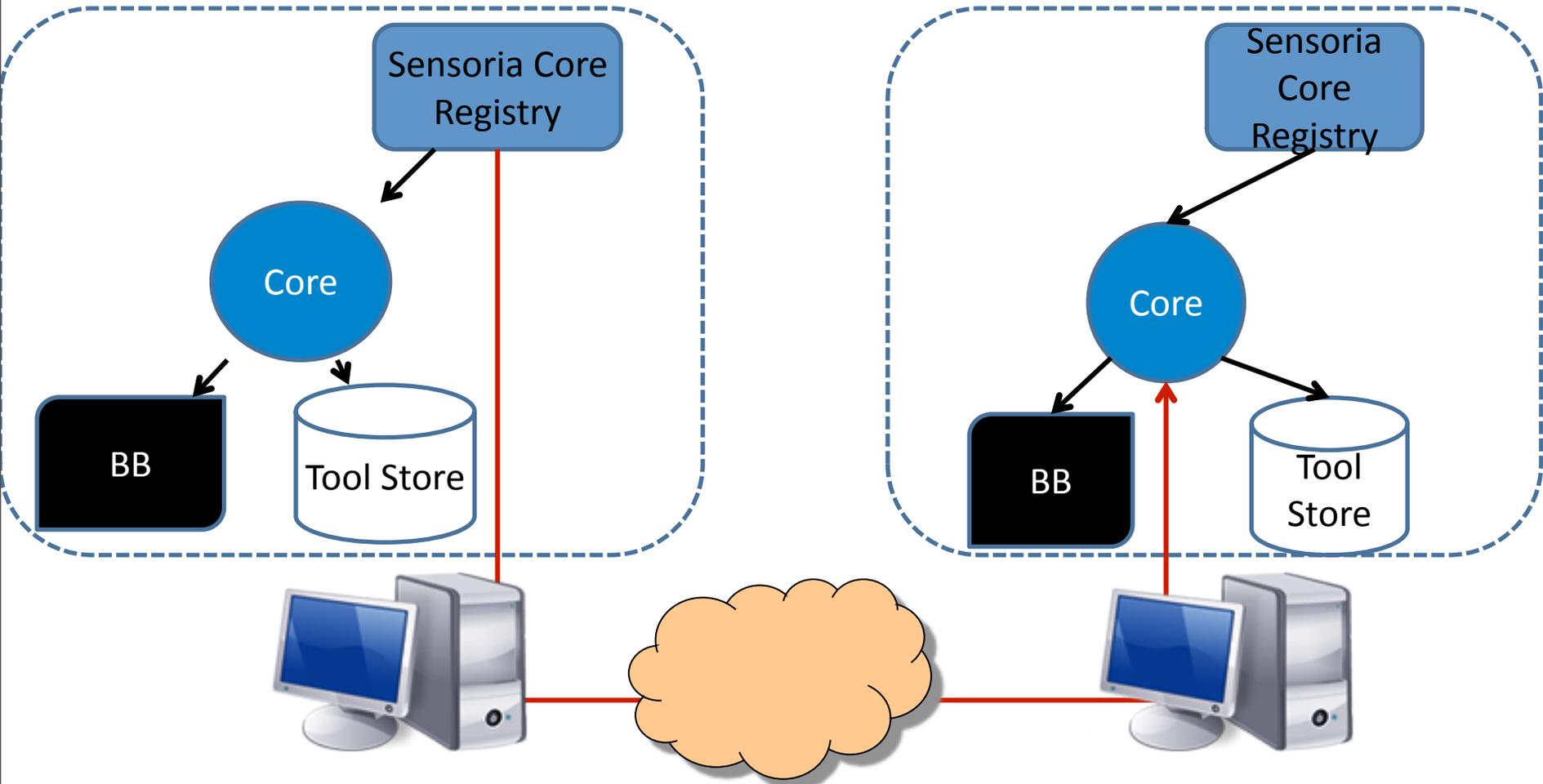
Sensoria



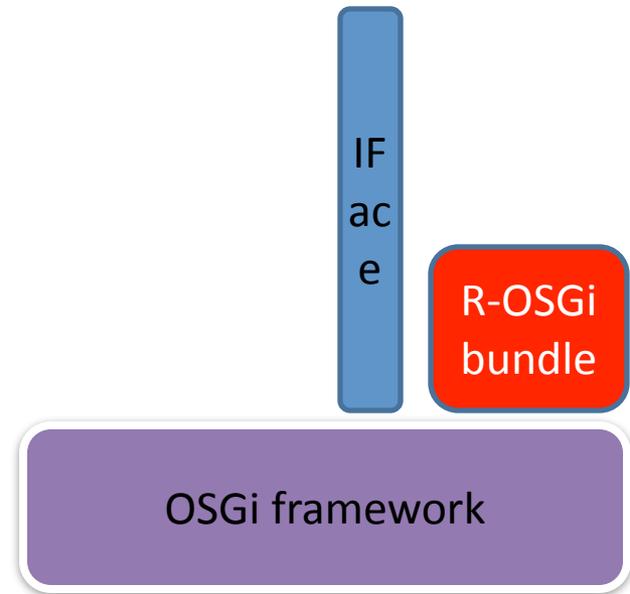
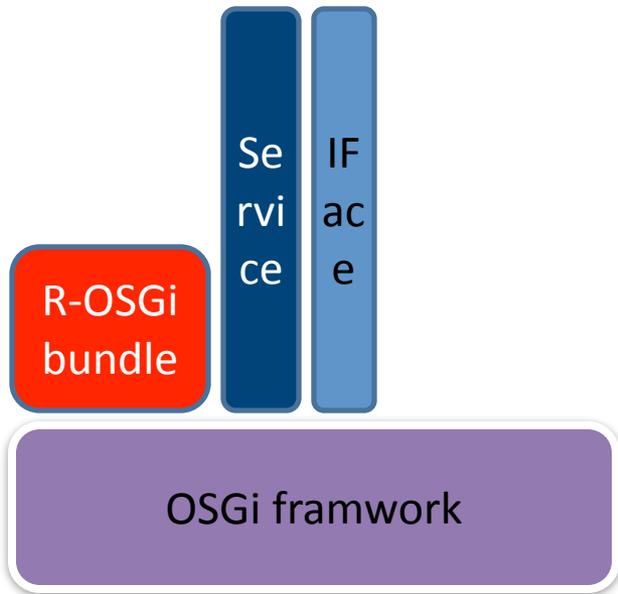
Sensoria



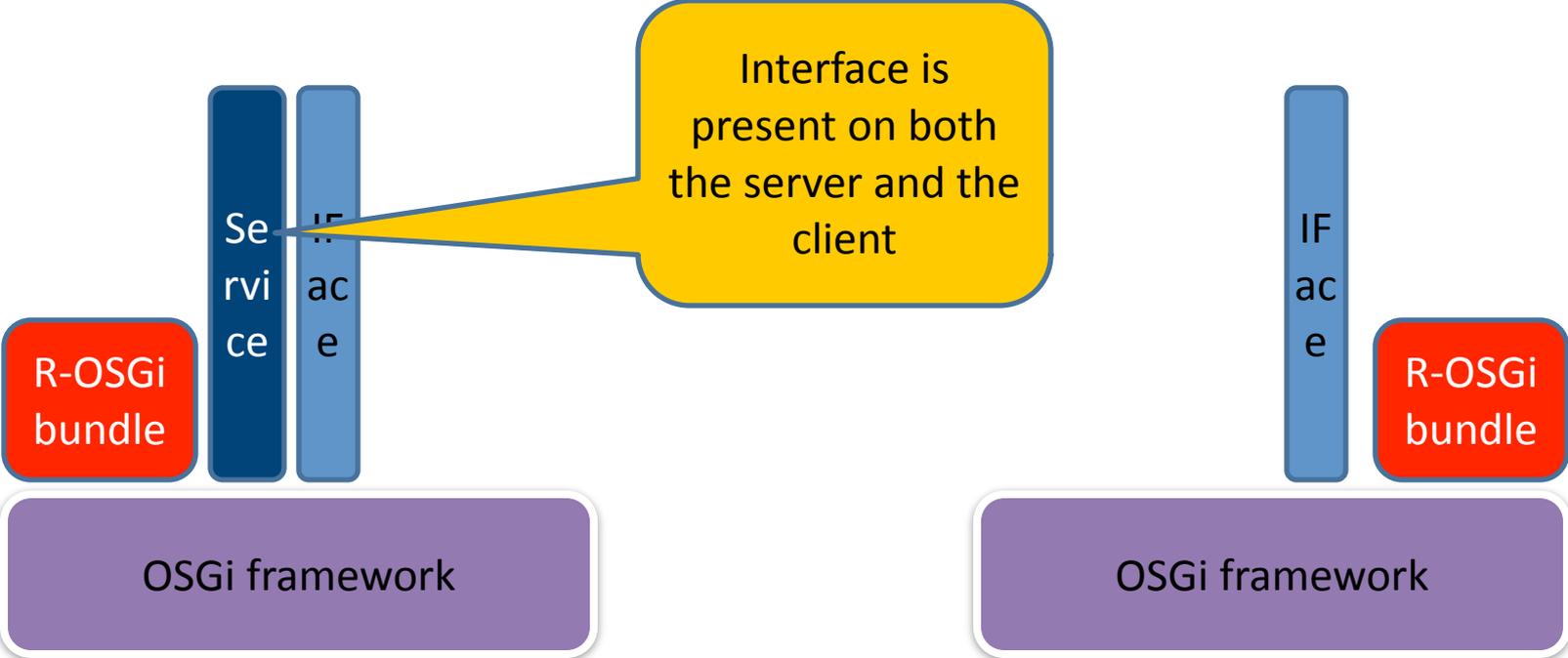
Distributed operation



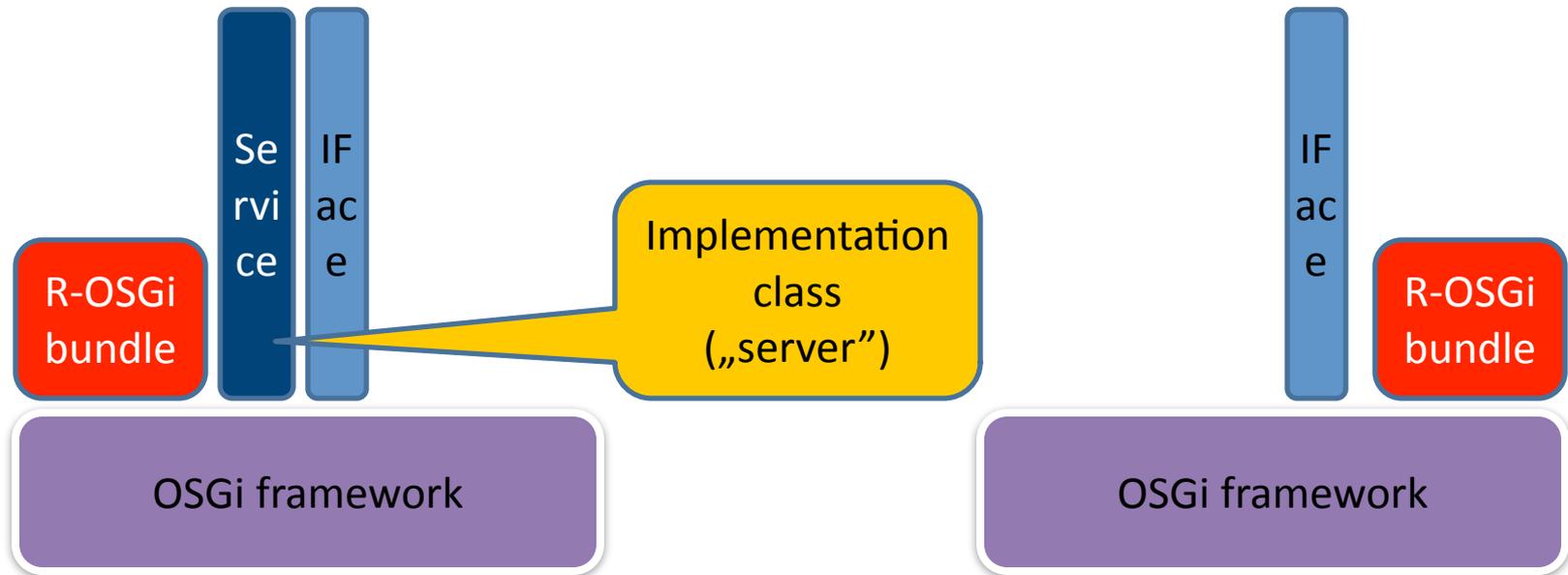
Remote OSGi (R-OSGi)



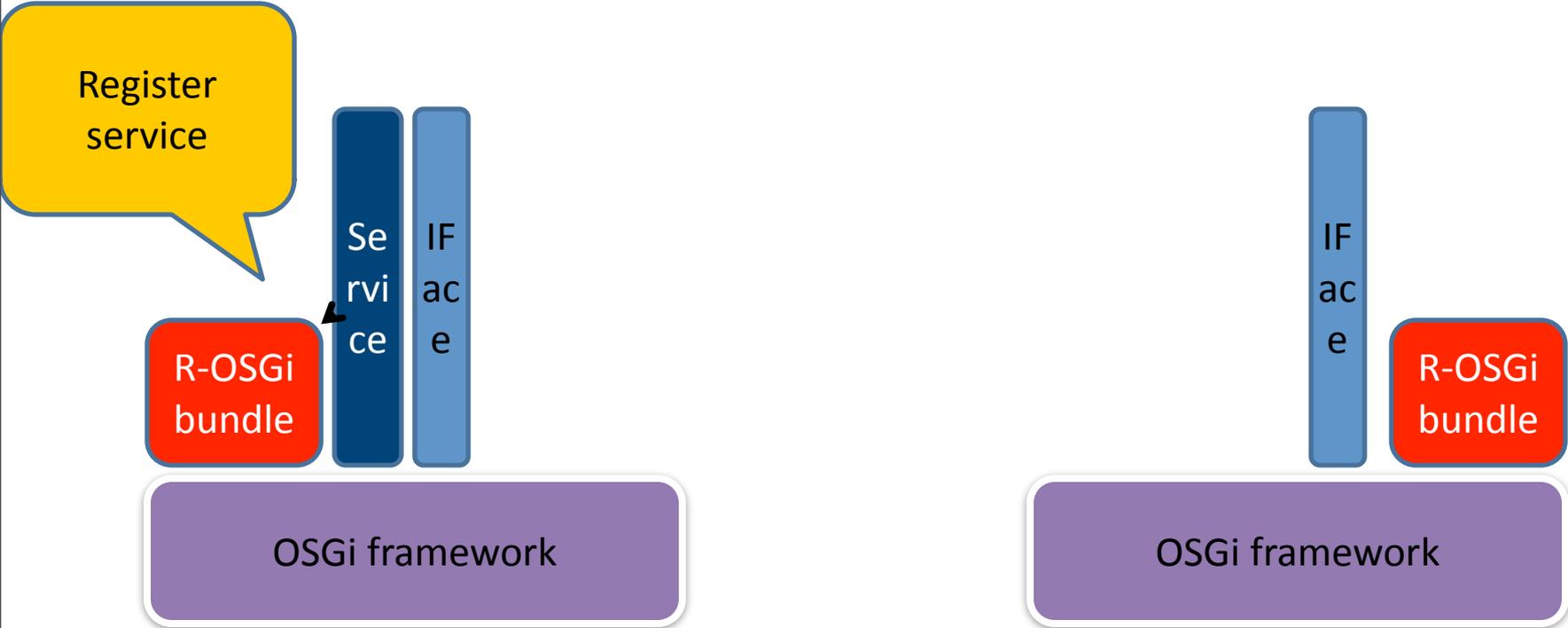
Remote OSGi (R-OSGi)



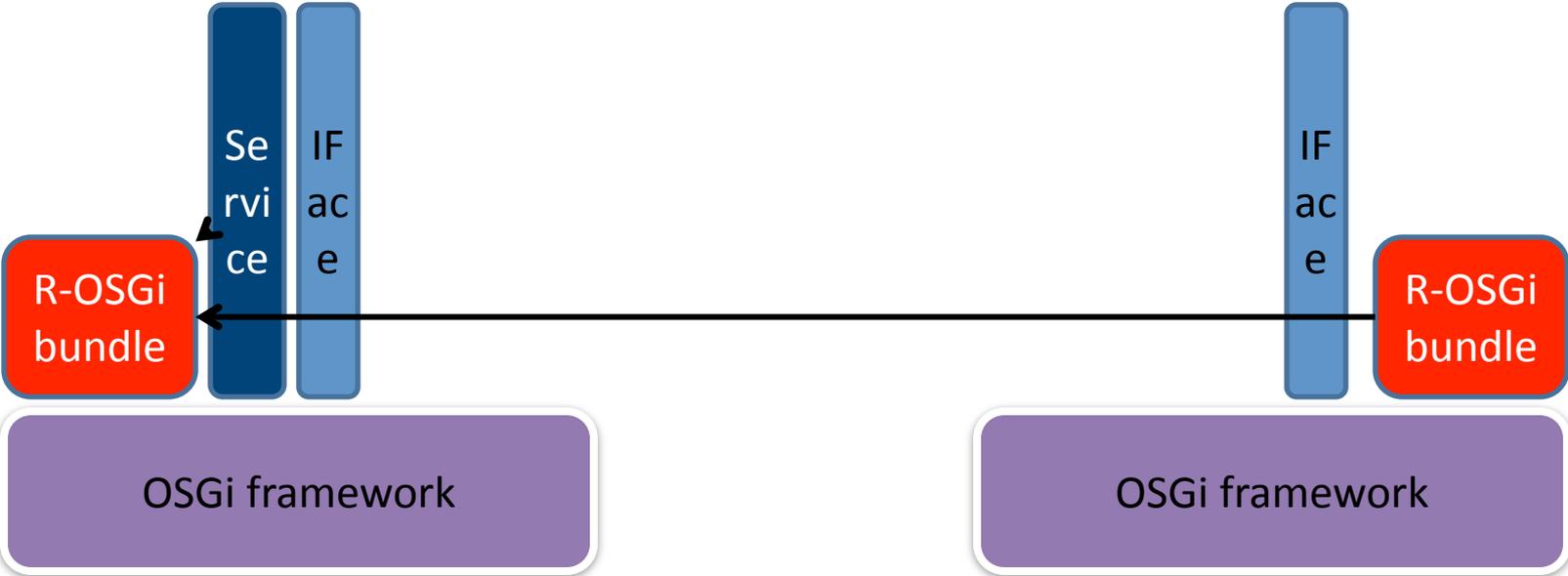
Remote OSGi (R-OSGi)



Remote OSGi (R-OSGi)

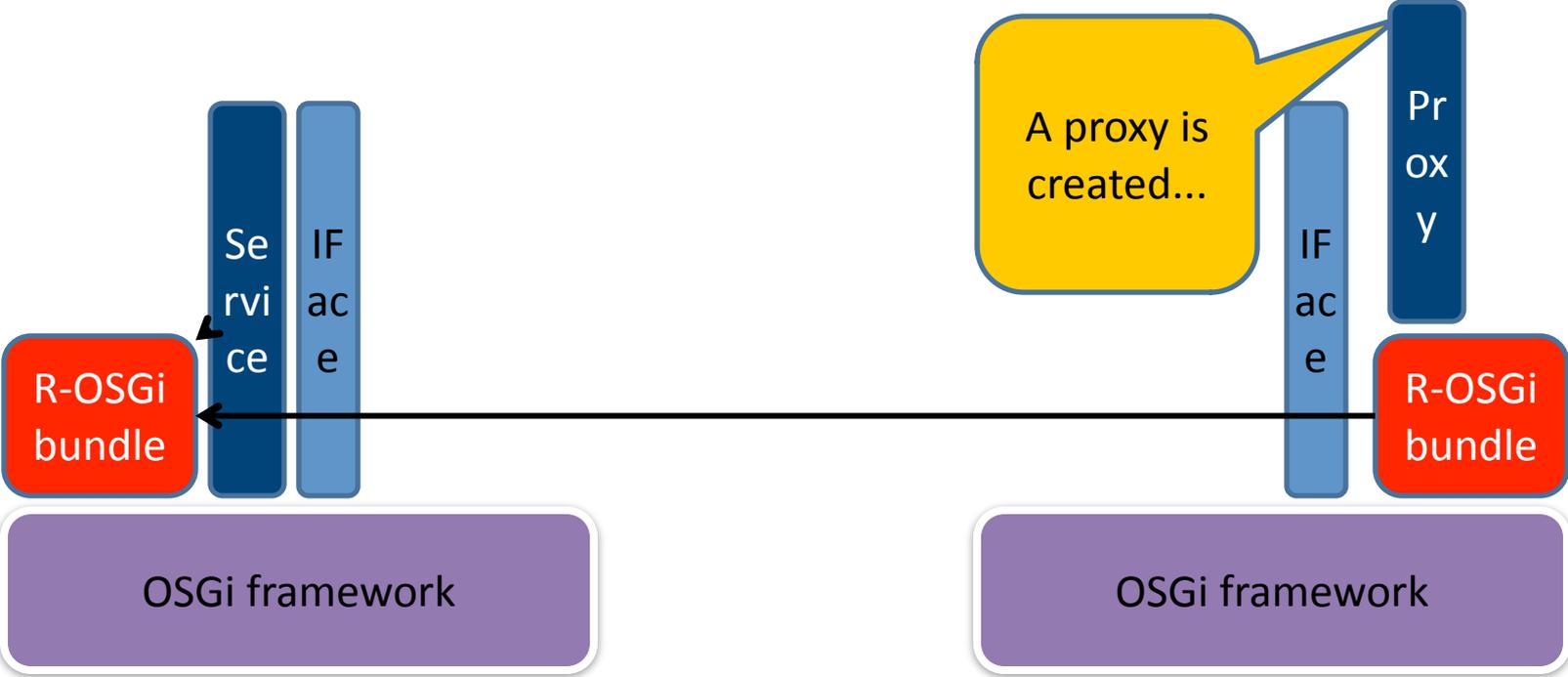


Remote OSGi (R-OSGi)

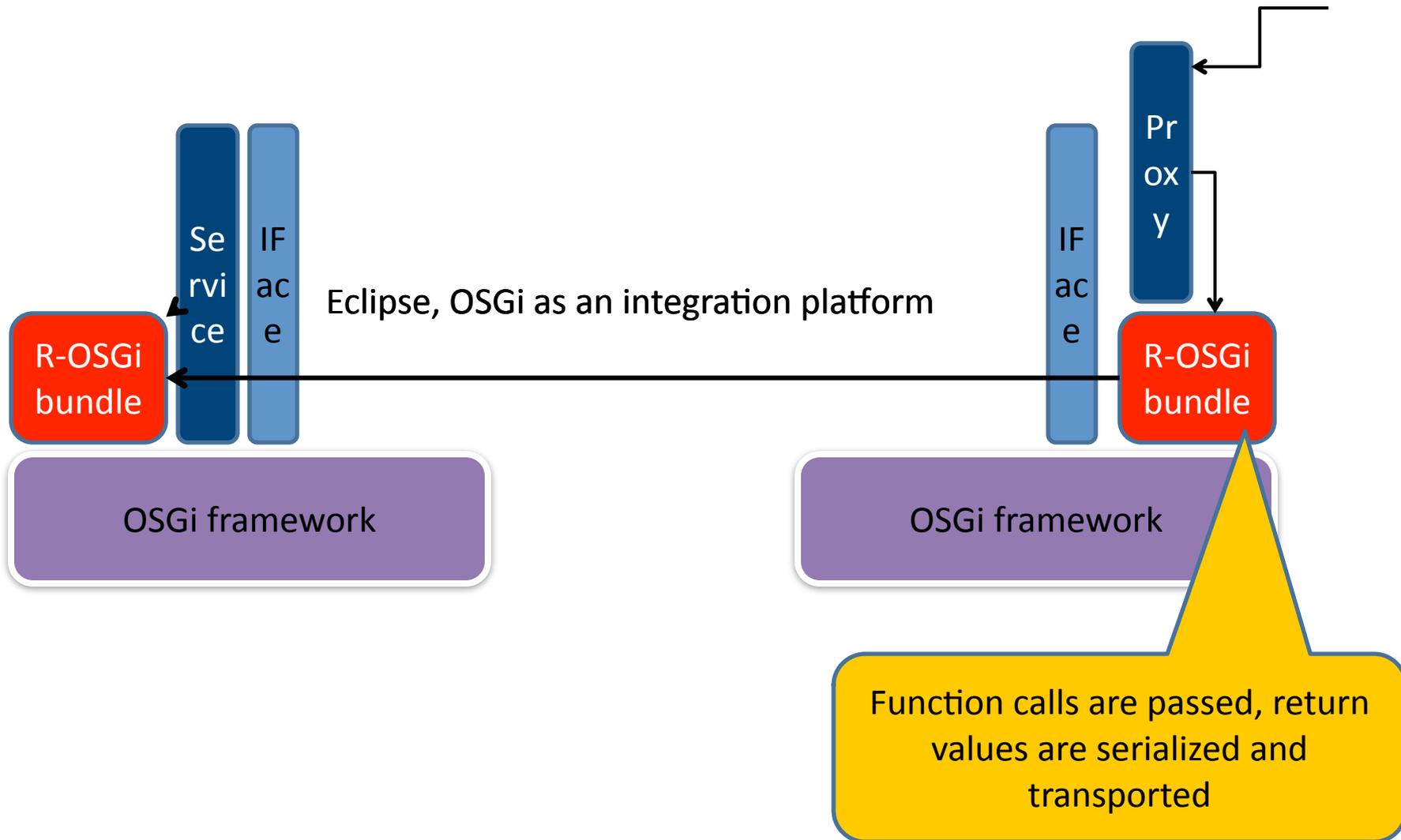


A remote client requests the service...

Remote OSGi (R-OSGi)



Remote OSGi (R-OSGi)



Comparing ECF and R-OSGi

- ECF
 - - More complex architecture
 - + More features (asynchronous calls, timing service, stb.)
- R-OSGi
 - + Simple
 - + Lightweight (only depends on OSGi)
 - Better suited for our goals

Workflow orchestration – Hello world

```
cReg.addRemoteCore("r-osgi://localhost:9278");  
  
remoteCore = cReg.findCoreByName("r-osgi://localhost:9278");  
  
testTool = remoteCore.findToolById("hu.bme.test.htmlconverter");  
  
testIf = testTool.getServiceInterface();  
  
sCore.print(testIf.htmlEscape("<<"));
```

Tool scripts

```
function soa2wsdl(umlFileName, outputDirectory)
{
    viatra = sCore.findToolByName("Viatra").getServiceInterface();
    vU = sCore.findToolByName("vUtil").getServiceInterface();
    fw = viatra.createFramework();

    vpml = vU.getInputStream(vU.getFileFromBundle
("hu.bme.mit.viatra.integration.sensoria", "model/
uml2soa_clean.vpml"));
    umlmodel_file = vU.getFile(umlFileName);
    viatra.nativeImportFromFile(fw, umlmodel_file, "uml");

    result = viatra.runTransformation
(fw, "transformations.uml2soa2wsdl", "Model=uml2.models."+umlmodel
+"_uml");
    viatra.writeoutput(fw, ...);
    viatra.disposeFramework(fw);
}
```

Outlooks

- Support a proper workflow orchestration engine instead of the built-in JS interpreter
 - Parallelism
 - Exception handling
 - Asynchronous messages
- UI integration
 - Allow for user interaction in automated workflows

Where to get it?

- General info:
 - <http://www.sensoria-ist.eu/>
- SDE:
 - <http://svn.pst.ifi.lmu.de/trac/sct>