Towards Compliance-driven Models, Languages, and Architectures for Service-oriented computing

Schahram Dustdar
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Vienna, Austria

http://www.infosys.tuwien.ac.at

Outline

- Introduction
- Problems in Compliance
- Proposed approach based on architectural views
- Conclusion and Future Work


TU
WIEN

COMPAS


TU
WIEN
Technische
Universität
Wien
Vienna
University of
Technology


Universität Stuttgart


Telcordia®


CWI


Tilburg University


THALES


LIRIS


UNIVERSITY
OF TRENTO - Italy
Information Engineering
and Computer Science Department


PRICEWATERHOUSECOOPERS


VITALAB
Distributed Systems Group

Targeted Field: SOC and SOA

- *Service-oriented computing (SOC)*
  - emerging computing paradigm
  - utilizes services as the basic constructs to support the rapid and easy composition of distributed applications
- *Service-Oriented Architecture (SOA)*
  - Architectural style in the field of Service-Oriented Computing
  - The service composition layer is typically on the top of a SOA
  - This layer often provides a process engine (or workflow engine) which invokes the SOA services to realize individual activities in the process
  - The main goal of such *process-driven SOAs* is to increase the productivity, efficiency, and flexibility of an organization via process management

COMPAS - Overview

- COMPAS addresses a major shortcoming in today's approach to design SOAs:
  - Throughout the architecture various compliance concerns must be considered
  - Examples*:*
    - Service composition policies, Service deployment policies,
    - Information sharing/exchange policies, Security policies, QoS policies,
    - Business policies, Jurisdictional policies, Preference rules Intellectual property and licenses
- So far, the SOA approach does not provide any clear technological strategy or concept of how to realize, enforce, or validate them

Compliance Concerns: Examples

- Service composition policies
  - e.g., use of data for a certain case only
- Service deployment policies
  - e.g., geographic restrictions for service instances
- Information sharing/exchange policies
  - e.g., request/response use specific message type
- Security policies
- QoS policies
- Business policies
- Jurisdictional policies
- Preference rules
- Intellectual property and licenses

Compliance Concerns Categories

- *Technical compliance concerns that must be validated at design time*
  e.g., compliance to composition policies

- *Technical compliance concerns that must be validated at runtime*
  e.g., compliance to QoS policies

- *Domain-oriented compliance concerns that must be validated at design time and runtime*
  - This category is the most complex one, as it is build on top of the two technical categories
  - *e.g., compliance to preference rules, to licenses, etc.*

Problem in Detail

- A number of approaches, such as business rules or composition concepts for services, have been proposed
- None of these approaches offers a unified approach with which all kinds of compliance rules can be tackled
- Compliance rules are often pervasive throughout the SOA
- They must be considered in all components of the SOA
- They must be considered at different development times, including analysis time, design time, and runtime

Current Practice for Dealing with Compliance

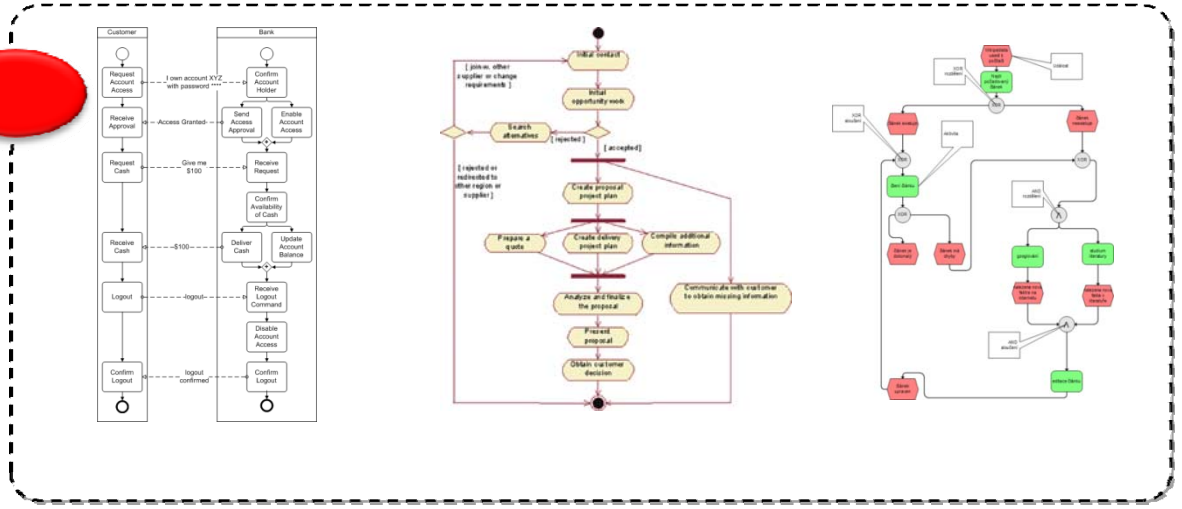
- In many cases, business compliance today is reached on a per-case basis
- Companies do not have a generic strategy for business compliance
- Instead they use ad hoc, hand-crafted solutions for specific rules to which they must comply

Issues with the Current Practice

- Systems are
  - hard to maintain
  - hard to evolve or change
  - hard to reuse
  - hard to understand
- It is difficult to ensure guaranteed compliance to a given set of rules and regulations
- It is difficult to keep up with constant changes in regulations and laws

Process-driven development scenarios

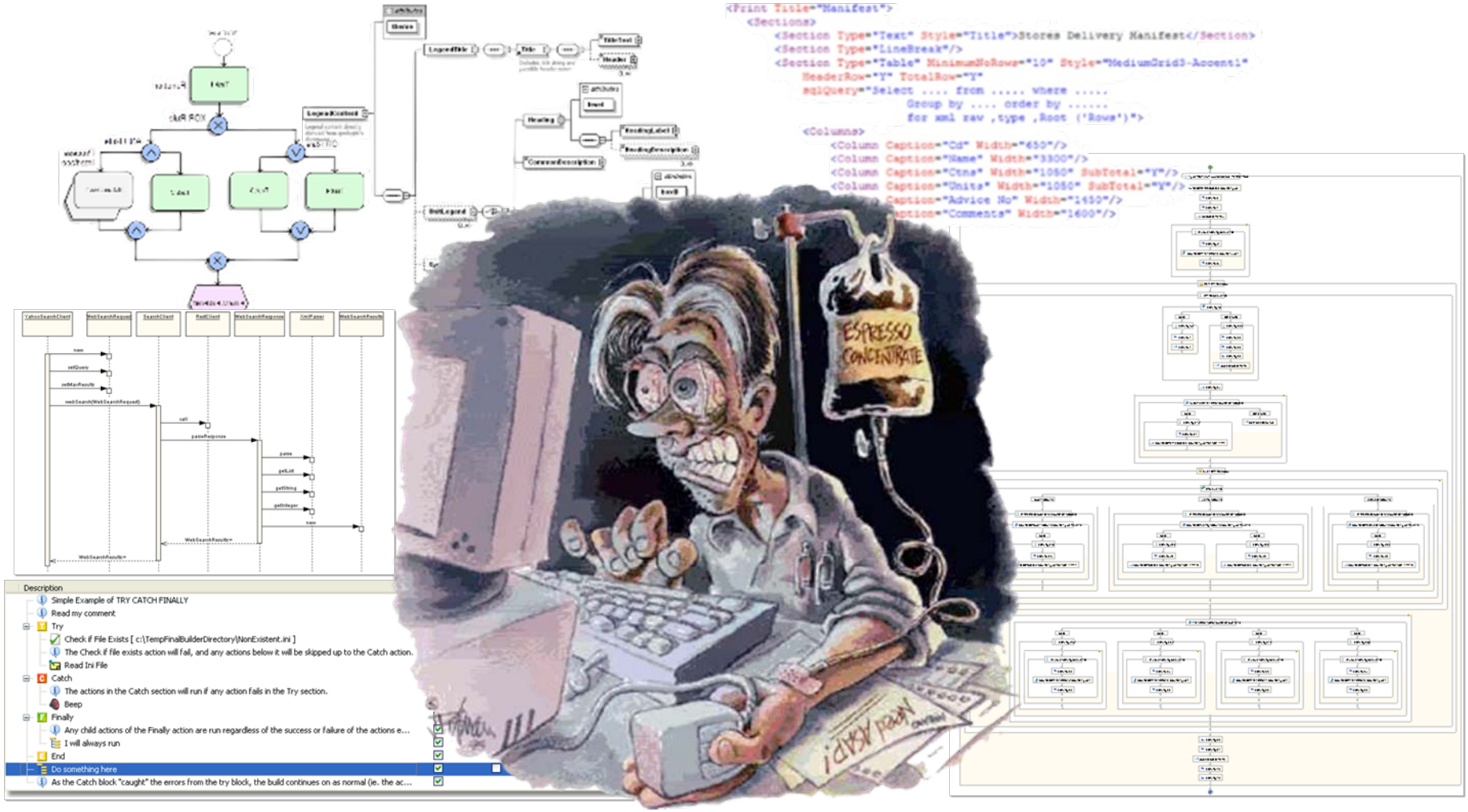
Business/domain experts



IT experts

```
142         byte[] result = XmlUtil.transformToBytes(node, "utf-8", true);
143         for (int ii = 0; ii < expResult.length; ii++) {
144             assertTrue(expResult[ii] == result[ii]);
145         }
146
147         expResult = ("<?xml version=\"1.0\" encoding=\"utf-8\"?>\r\n" +
148                 "<helloWorld xmlns=\"urn:Foo\"> Hello World!!!\r\n" +
149                 "  <string>input0</string>\r\n" +
150                 "  <byte>a</byte>\r\n" +
151                 "  <short>5</short>\r\n" +
152                 "  <int>12345</int>\r\n" +
153                 "  <long>12345</long>\r\n" +
154                 "  <float>2.22</float>\r\n" +
155                 "</helloWorld>\r\n").getBytes();
156         result = XmlUtil.transformToBytes(node, "utf-8", false);
157         for (int ii = 0; ii < expResult.length; ii++) {
158             assertTrue(expResult[ii] == result[ii]);
159         }
160
161         result = XmlUtil.transformToBytes(node, "utf-8", false, "xml");
162         for (int ii = 0; ii < expResult.length; ii++) {
163             assertTrue(expResult[ii] == result[ii]);
164         }
165
166         expResult = ("Hello World!!!").getBytes();
167         node = document.createTextNode("Hello World!!!");
168
169         result = XmlUtil.transformToBytes(node, "utf-8", false, "text");
170         for (int ii = 0; ii < expResult.length; ii++) {
171             assertTrue(expResult[ii] == result[ii]);
172         }
173
174         System.out.println("Successfully tested transformToBytes");
175     }
```

Process-driven development scenarios



Developer's nightmare – numerous tangled process concerns

Motivation: interoperability and reusability of processes

- Why process description (re)use is difficult
  - *The integration of many tangled aspects hinders understandability, modularity*
    - the control flow, service interactions, message and message types, fault handling, transactions, compliances, process engine configurations, etc.
  - *Stakeholders have different point of views, abstraction levels, skill sets, needs, etc.*
    - Business or domain experts
    - IT experts: developers, administrators

Existing solutions so far?

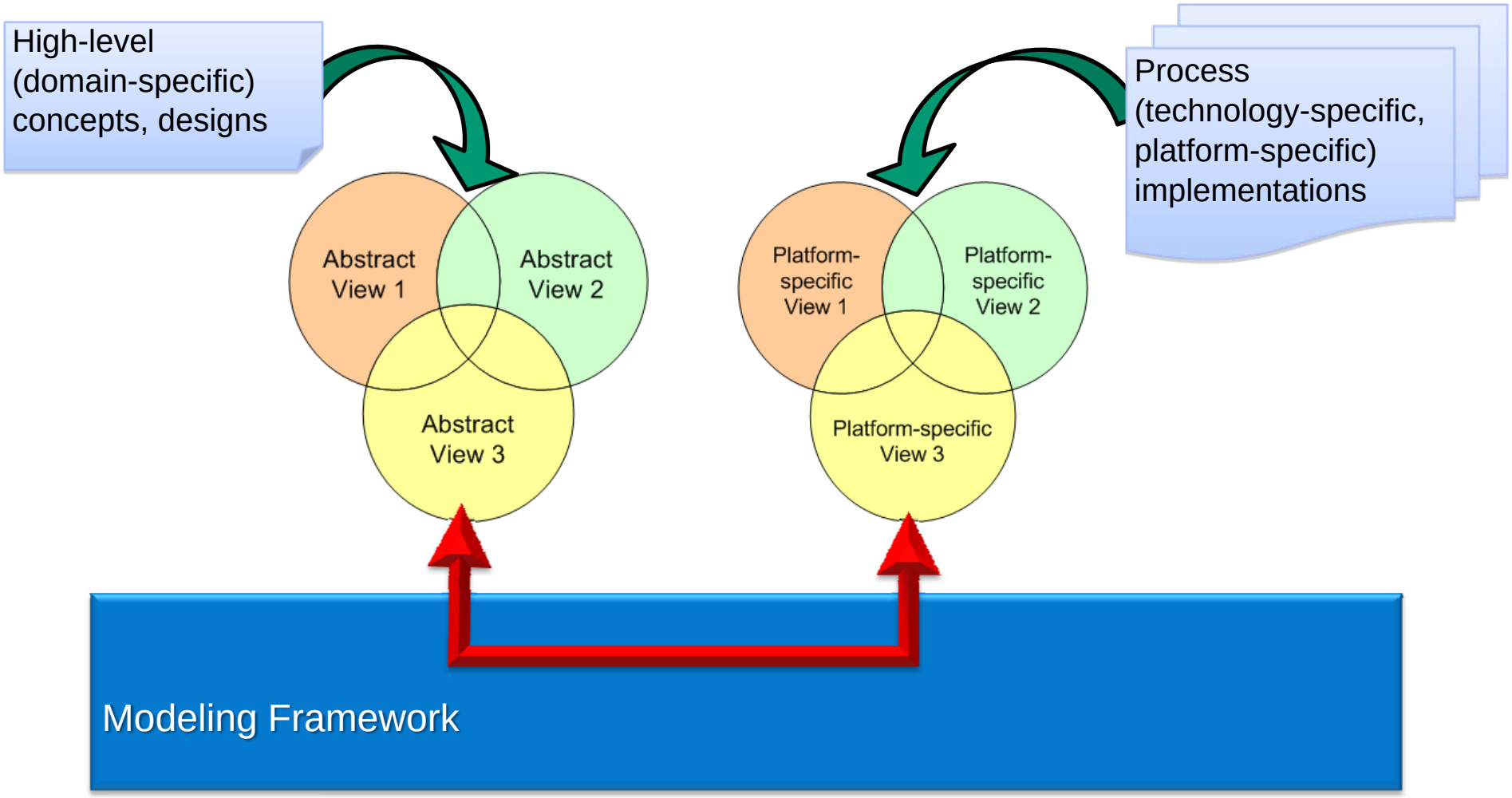
- Manually done by ”**copy and paste**“
  - Time-consuming and error-prone
  - Only IT developers are able to do “copy and paste“
  - Hinders scalability and agility
- Language transformation-based approaches
  - E.g., BPMN/EPC/UML Activity diagrams to BPEL/WSDL, and vice versa.

- **Workflow** mining
  - Extracts (only) workflows from applications

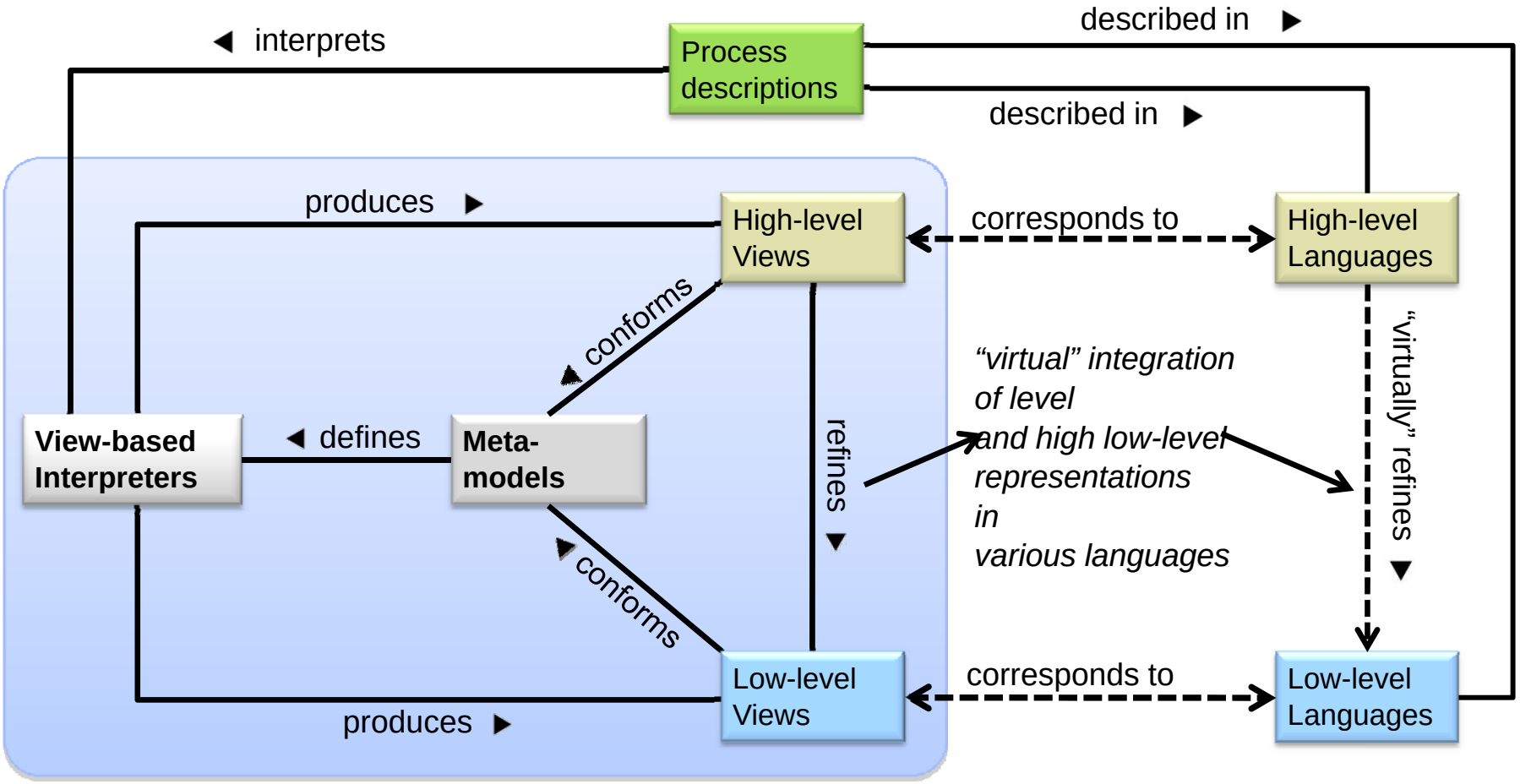
Proposed solution: View-based Model-driven Engineering

- Separation of concerns principle
  - A realization: concept of **architectural views**
- Model-driven engineering
  - **(semi)-formalization of process fragments** to enhance modularity, reusability, etc.
  - separation of abstraction levels by tailored views to enhance adaptability, understandability
- **View-based reverse engineering**
  - extracts (semi)-formalized views

Proposed integration solution

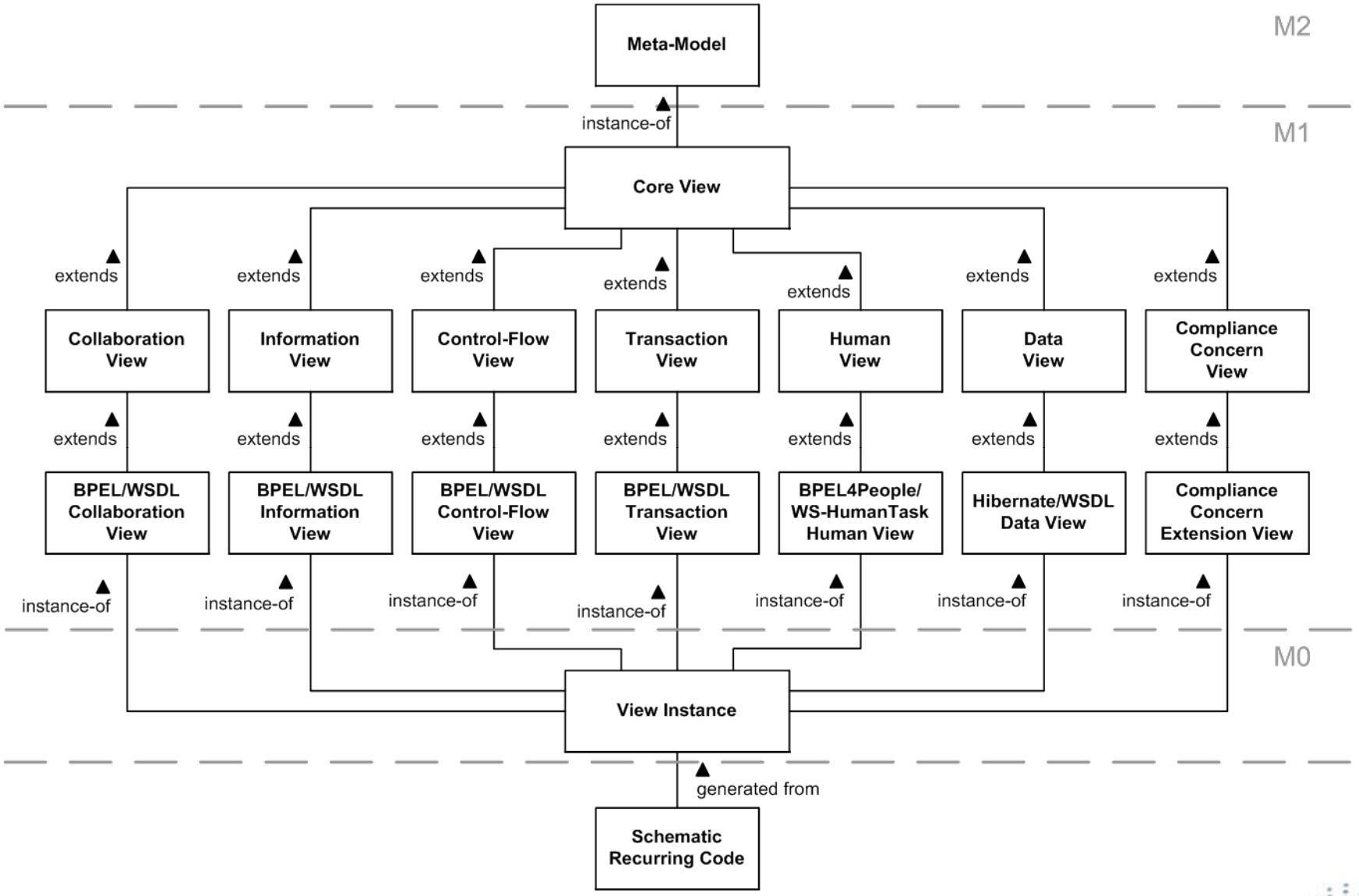


View-based integration architecture

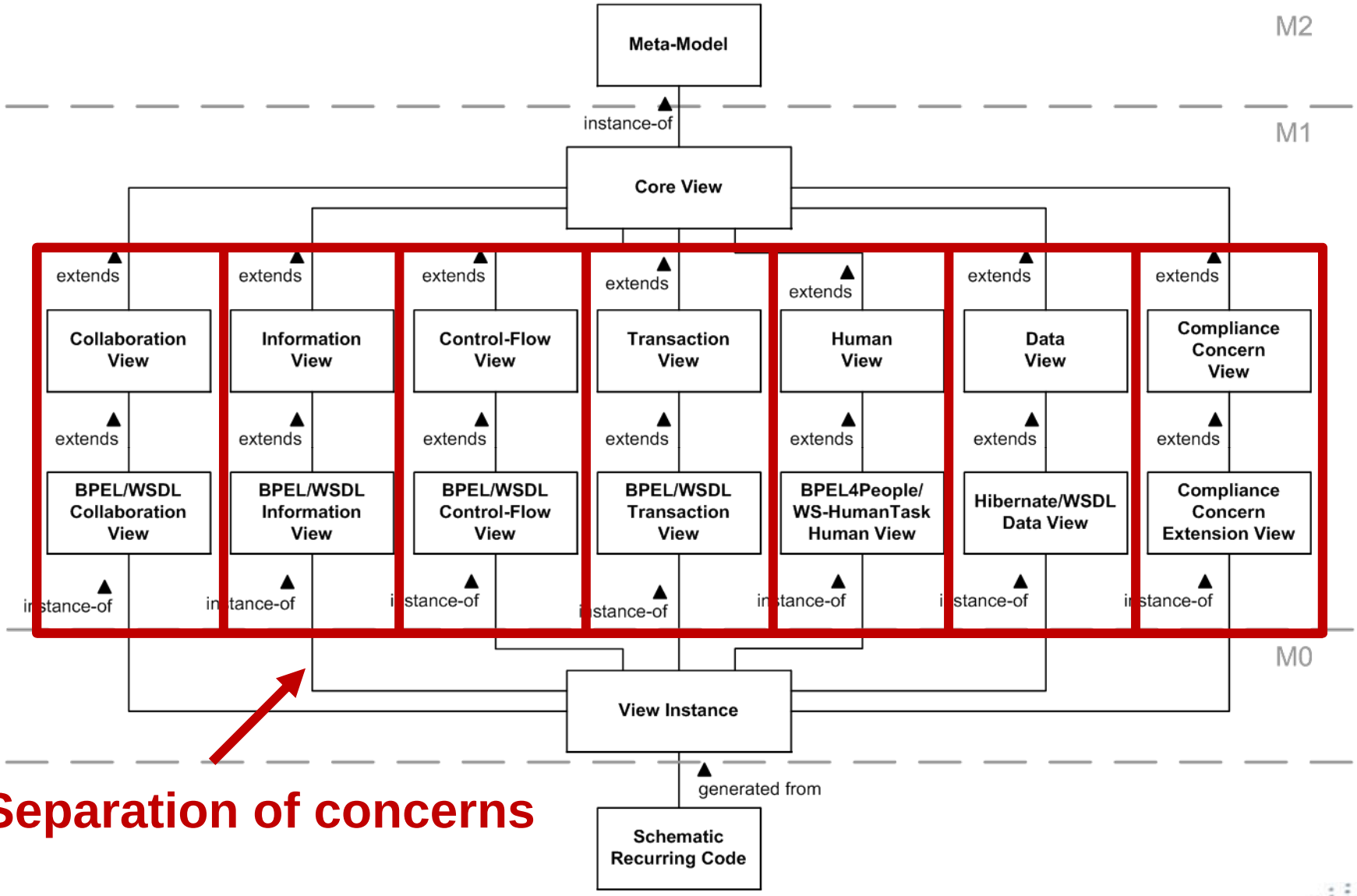


View-based Modeling Framework

View-based Modeling Framework (VbMF)

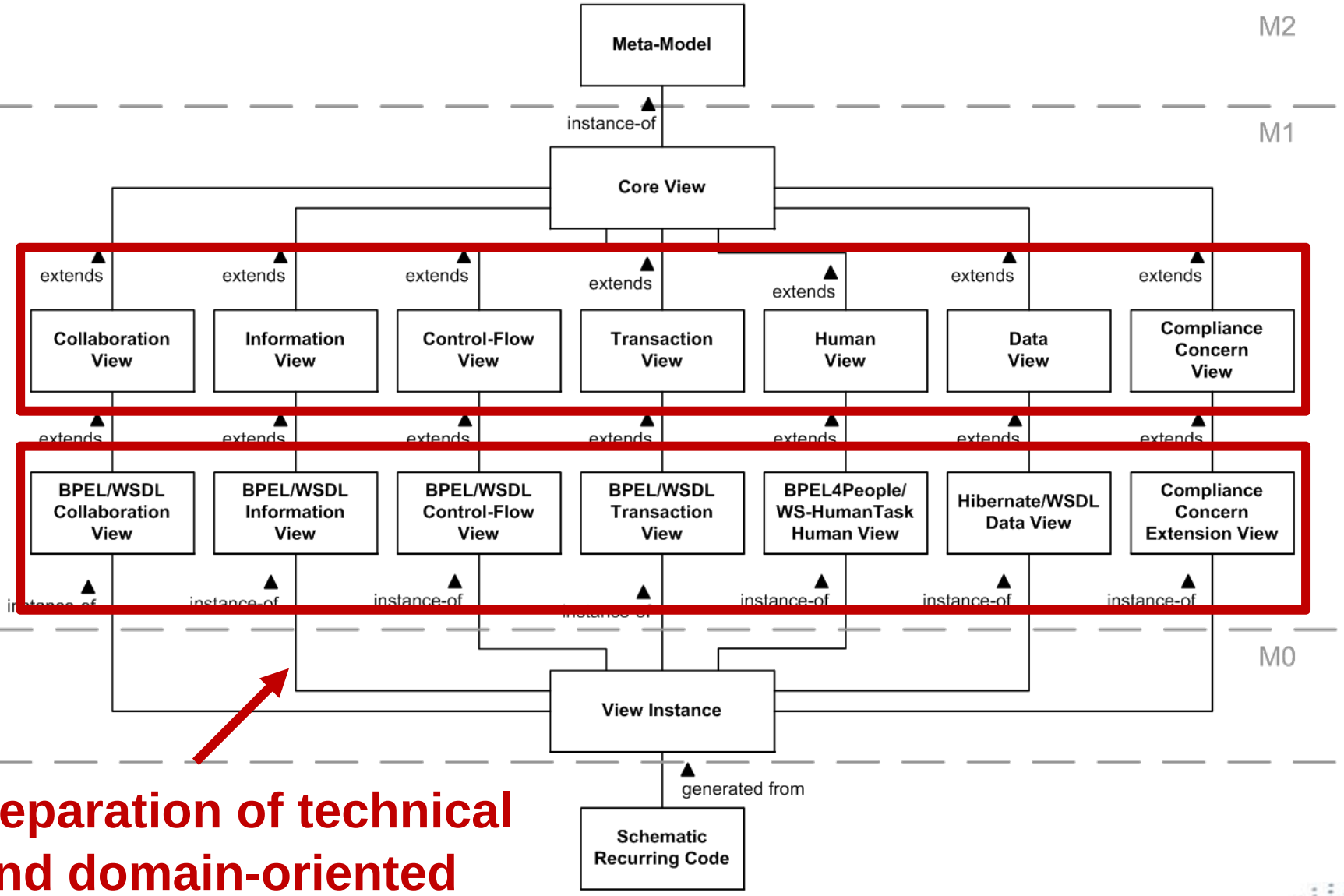


View-based Modeling Framework (VbMF)



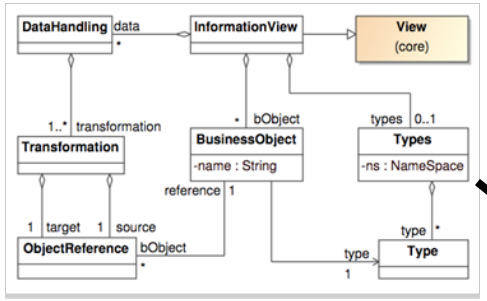
Separation of concerns

View-based Modeling Framework (VbMF)



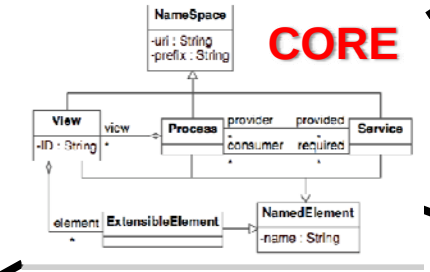
Separation of technical and domain-oriented views

View-based Modeling Framework (VbMF)

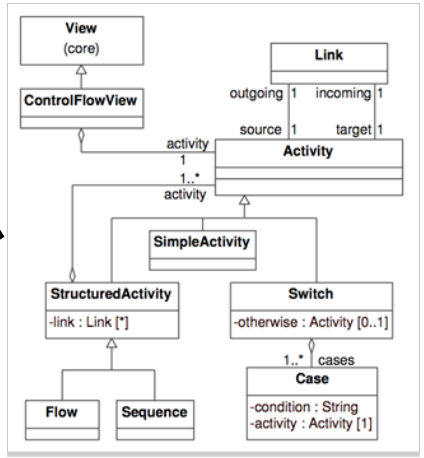


Information View meta-model

extends

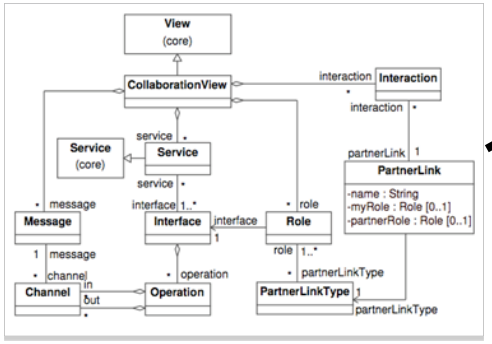


extends



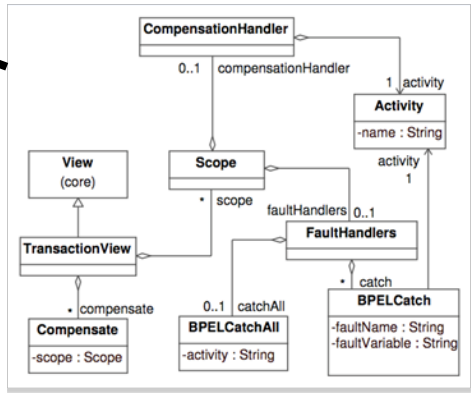
Control-flow View meta-model

extends



Collaboration View meta-model

extends

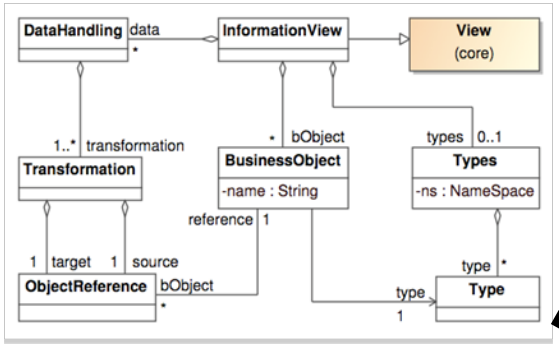


Transaction View meta-model

Horizontal extensibility of VbMF

VITALAB Distributed Systems Group

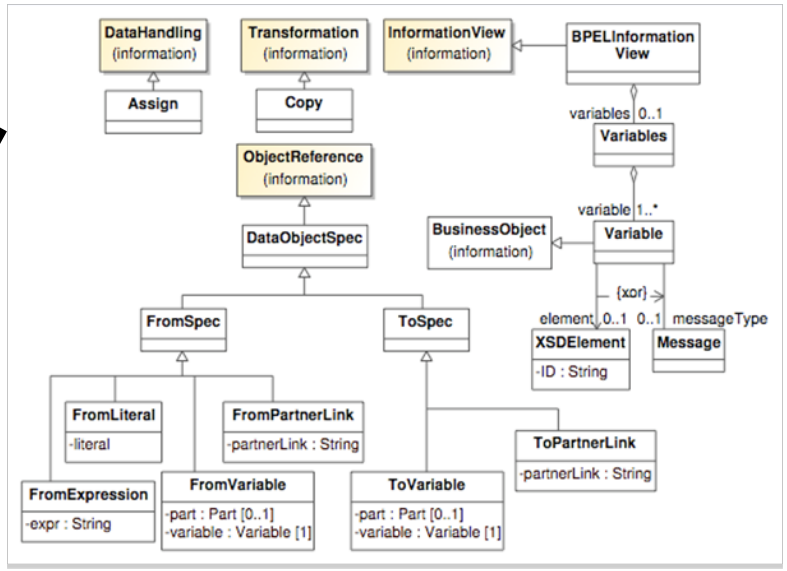
View-based Modeling Framework (VbMF)



Information View meta-model

extends

An extension of the Information View meta-model

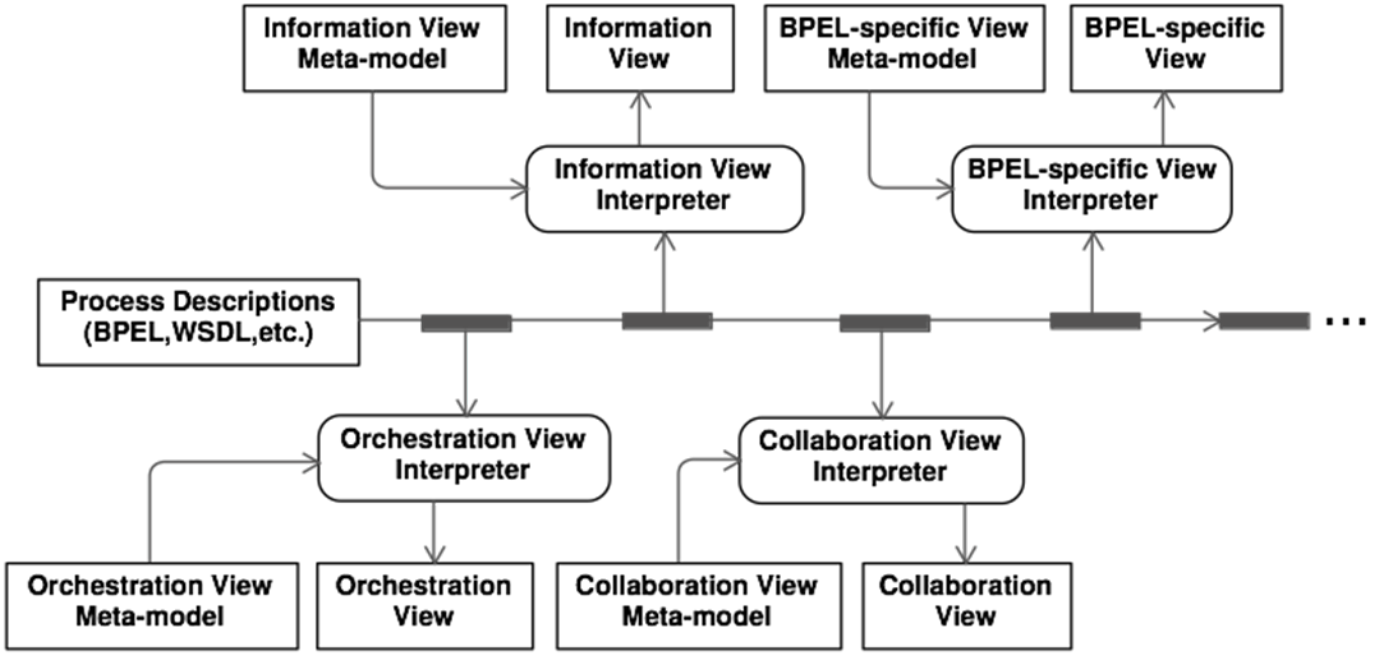


High-level abstraction

Lower-level abstraction

Vertical extensibility of VbMF

View-based Interpreters



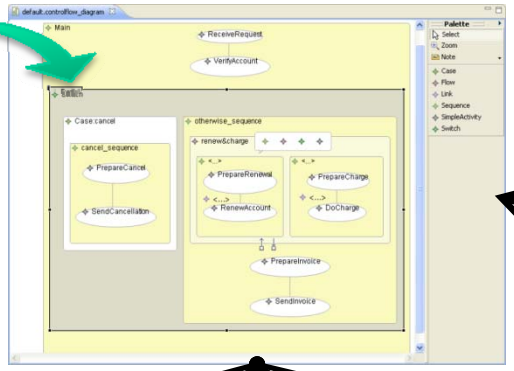
Chain of Responsibility and *Partial Interpreter Pattern* - General Approach For View Extraction

The overall Tool chain

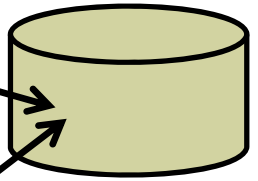
- Design
  - Modeling of Views within the VbMF (EMF/GMF-Editor)
- Transformation
  - Invocation of the Code Generation (oAW-Workflow)
- Validation
  - Semantic validation/optimization of a process
- Deployment
  - Prepare and deploy process on a BPEL engine
- Execution
  - Fire & Forget a long-running process

How views are reused?

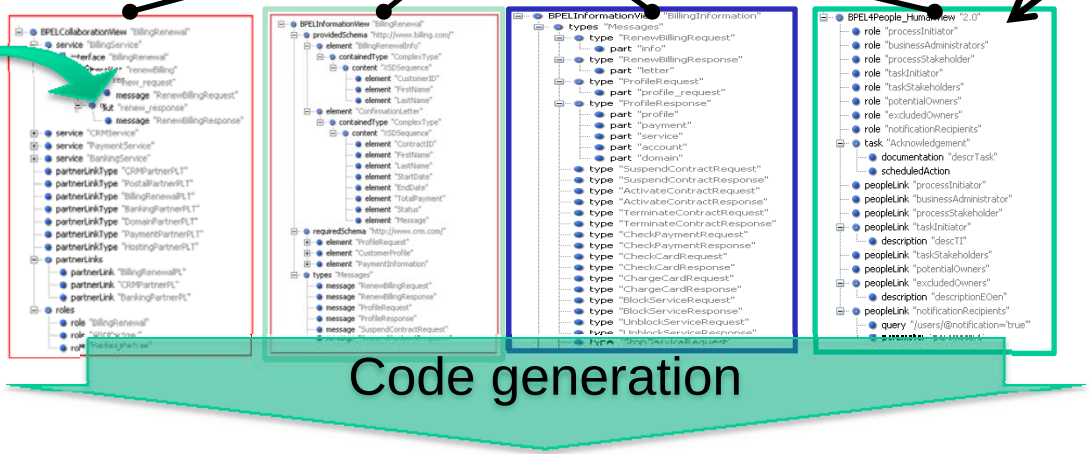
Business, domain experts



High-level, problem-oriented views



View-based repository



Low-level, technology speficic Views

IT experts

executable code, configurations

Current Work

- Collaborative Model-Driven Development
  - Lightweight Collaborative Model-Driven Environment
  - Correlation of Process Stakeholders & MDD Artefacts
  - Model Repository
- Distributed Process Monitor
  - for Debugging, Logging, Monitoring, etc.
  - Publishers = Components
  - Broker Architecture
  - Distributed Subscribers

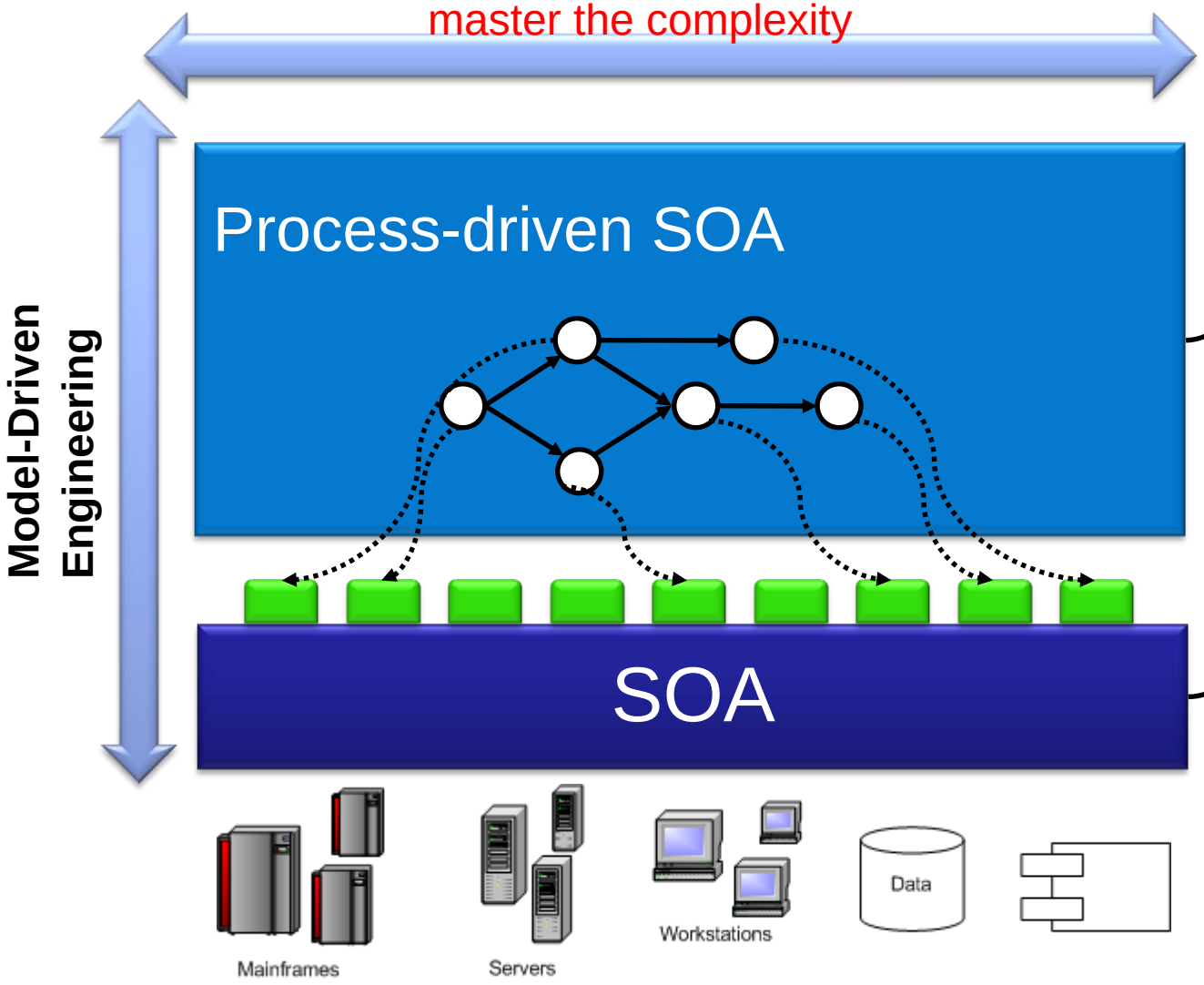
Current Work

- Dependency management
  - Traceability
  - Change impact analysis
  - Change propagation
- Collaborative view repository
  - View-based, model-driven repository

Summary of our approach

VbMF

Separation of Concerns
(e.g., *architectural views*) to
master the complexity



provides an efficient method for integrating business functionality

reconciles the heterogeneous nature of software systems

VITALAB Distributed Systems Group

Thanks for your attention!

Schahram Dustdar

Distributed Systems Group,

Institute of Information Systems,
Vienna University of Technology, Austria

http://www.infosys.tuwien.ac.at