FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Rigorosum — 19 January 2011, 13h
Zemanek Seminarraum

Supporting Model-Based Reflection, Monitoring, and Evolution in Service-Oriented Architectures through Model-Aware Systems

Candidate: Ta'id HOLMES

Chair: Prof. Gerti KAPPEL

Advisor: Prof. Schahram DUSTDAR

External Examiner: Prof. Uwe ZDUN

VITALAB Vienna Internet Technologies Advanced Research Lab
Distributed Systems Group

Context and Background

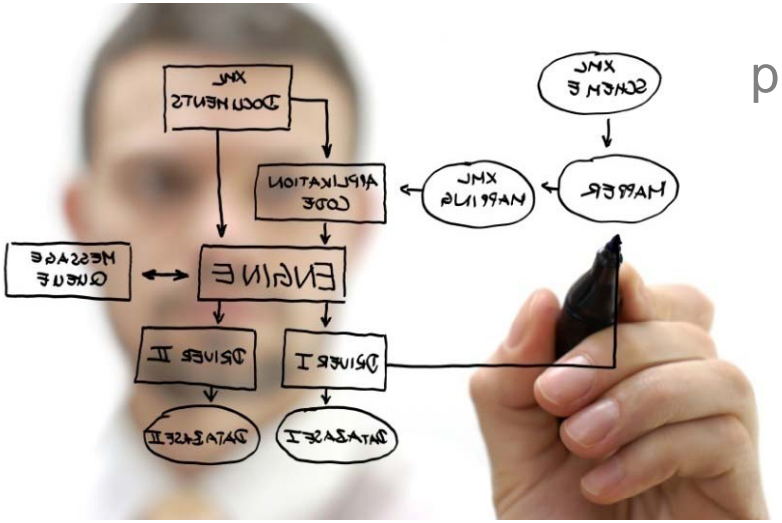
software systems become increasingly **complex**

- unify different technologies
- are adapted for new and emerging technologies
- need to comply with imposing requirements

Model-Driven Engineering (MDE)

- helps to master complexity (design-time)
- utilizes **models** as central artifacts

MDE Artifacts



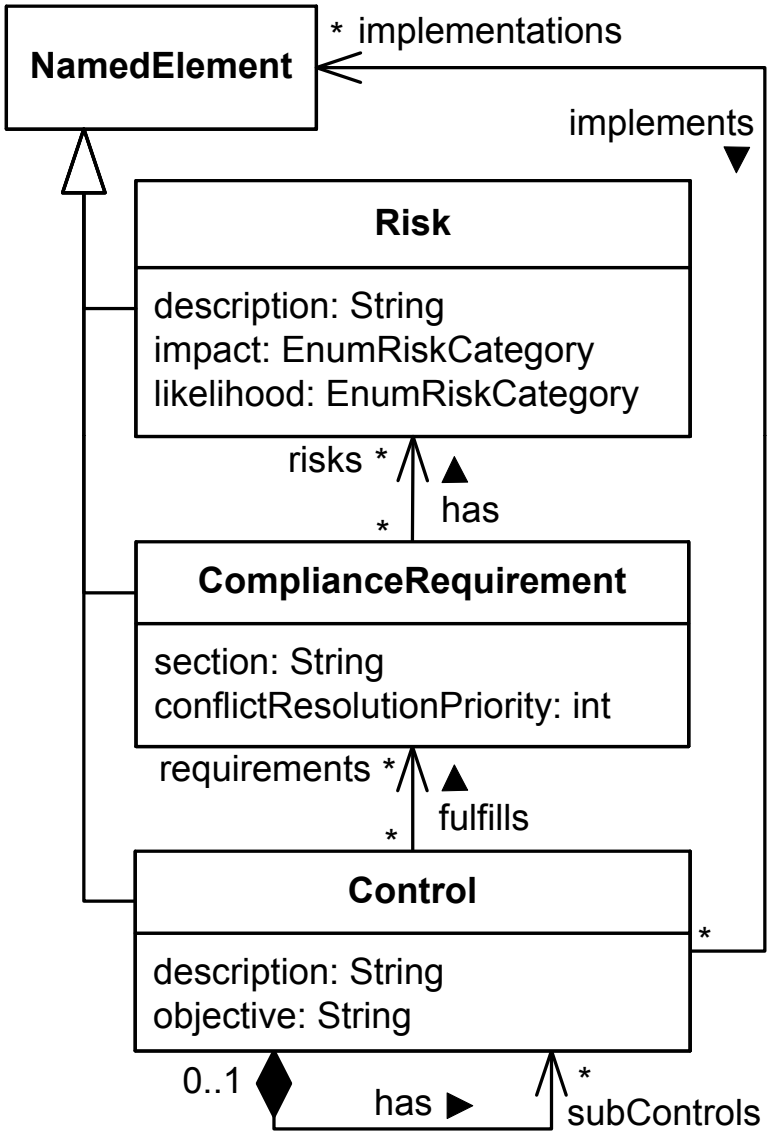
Models

- precisely specified
- instances can be validated
- can be (d|r)efined at different abstraction levels
- are suiteable to be represented to stakeholders
- can be bound to tailored DSLs

Model Transformations

- capture technical expertise (e.g., PIM → PSM)
  ⇒ eases portability & adaptation
- generation step: model to code transformation
  (recurring) code ⇒ eases maintenance

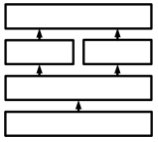
MDE Artifacts — Examples




```
<h2>Risk-Control Correlation Matrix</h2>
<table>
  <tr>
    <th>Risks/Controls</th>
«FOREACH cv.control AS c»
    <th>
      <a href="«cv.processName+
        "_C_"+c.uuid+".html"»">«c.name»</a>
    </th>
«ENDFOREACH»
  </tr>
«FOREACH cv.risk AS r»
  <tr>
    <th>
      <a href="«cv.processName+
        "_R_"+r.uuid+".html"»">«r.name»</a>
    </th>
  «FOREACH cv.control AS c»
    <td>
    «IF (c.requirements.risks.contains(r))»
      X
    «ENDIF»
    </td>
  «ENDFOREACH»
  </tr>
«ENDFOREACH»
</table>
```

MDE — Problems

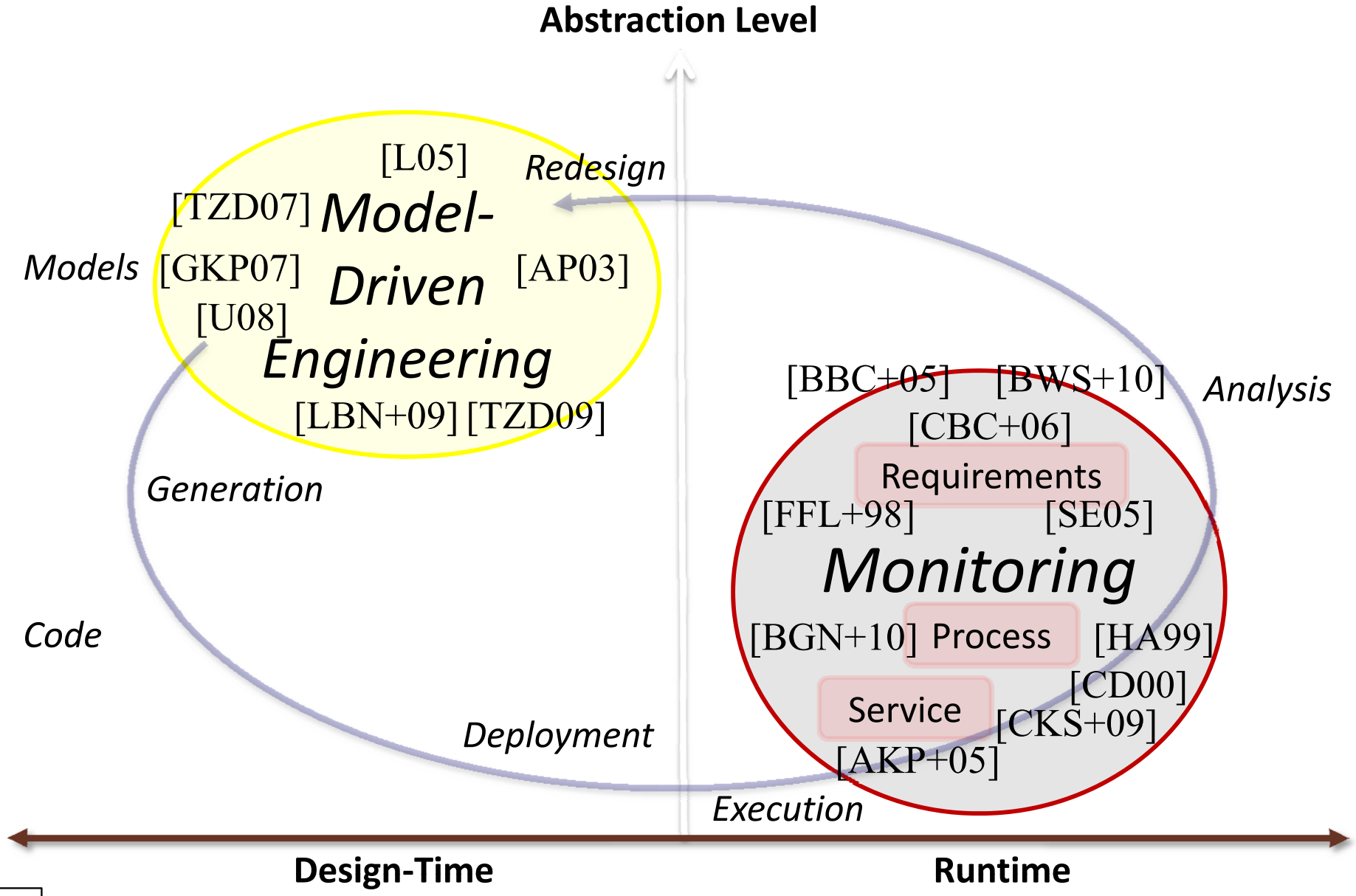
- evolution and co-evolution of MDE artifacts and systems
- concurrent work
  - few MDE tools offer collaboration support ⇒ lack of integration
  - common version control systems are too naïve for MDE
    - ⇒ versioning on a model element level is not supported
    - ⇒ relationships between artifacts are not captured/managed
- search & retrieval of models and MDE artifacts
  - missing tool support and infrastructures ⇒ reuse becomes difficult
- traceability (high-level ↔ low-level model-instances and code)
  - essential for meaningful feedback from the runtime to stakeholders and for identifying and understanding the root-cause
- generation step causes different MDE phases to be isolated
  - missing infrastructure that supports dynamic model look-up ⇒ model-based reflection is rarely used
- monitoring of model-driven systems (e.g., in regard to requirements)



Addressing MDE Challenges

- Support for *various* stakeholders
  - appropriate model-representations (DSLs)
  - role-based access controls (RBACs)
- Dealing with *concurrency*
  - locking mechanisms
  - raising the awareness of the work of others
  - comparing and merging possibilities
  - support for resolving conflicts
- Management of *MDE Projects & Artifacts*
  - versioning
  - capturing and keeping track of relationships
  - support for model evolution
- Support for model-aware *entities*
  - information retrieval services
  - resource management services

MDE Lifecycle — Positioning



Research Question 1

Artifacts are subject to change & evolution of meta-models may require co-evolution of artifacts and systems if navigability is affected:

How can navigation incompatibilities be detected?

Research Question 2

During execution, systems could benefit from reflecting on models:

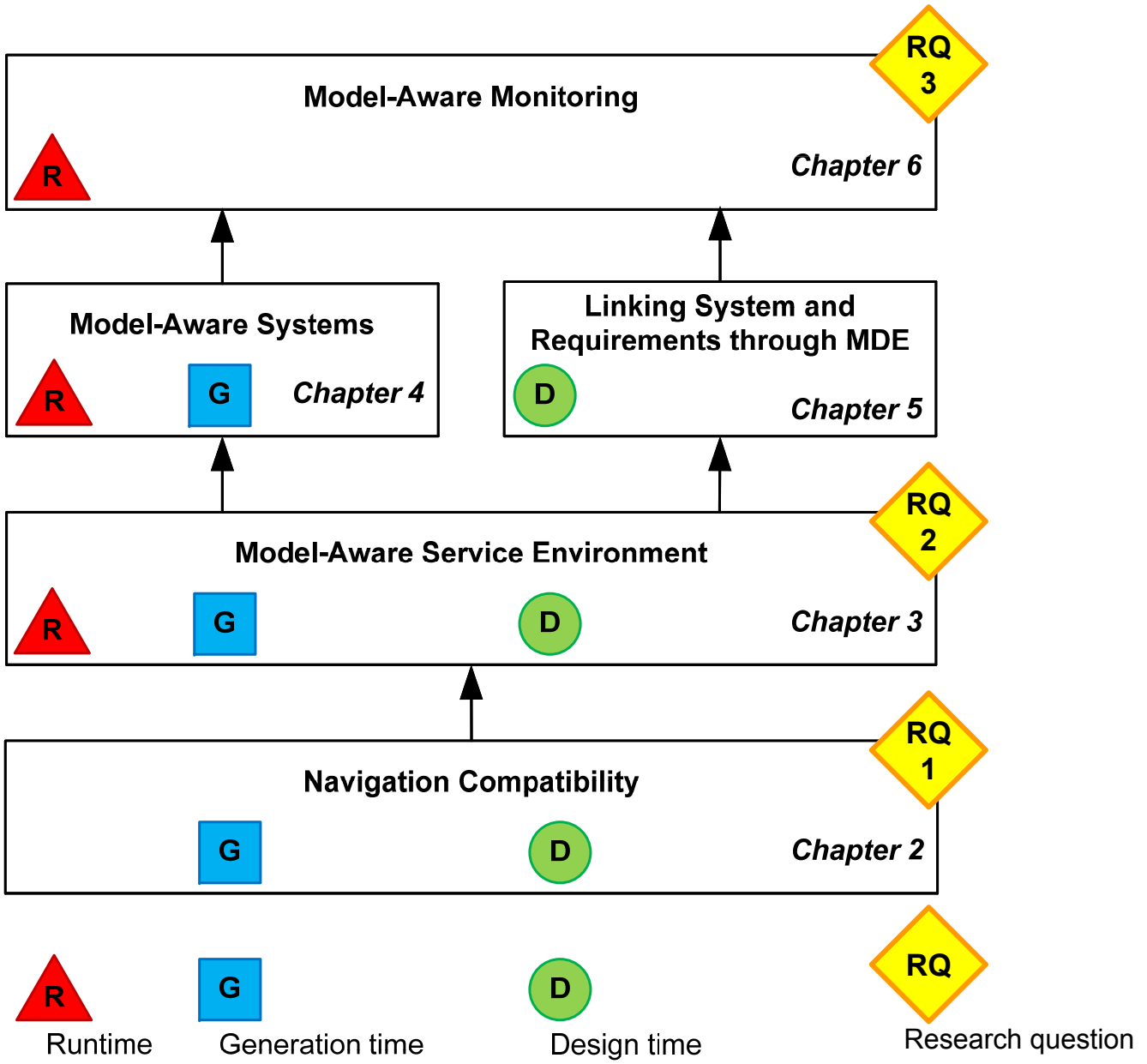
How can MDE projects and artifacts be retrieved, searched for, and managed both at design time and runtime in a distributed setting?

Research Question 3

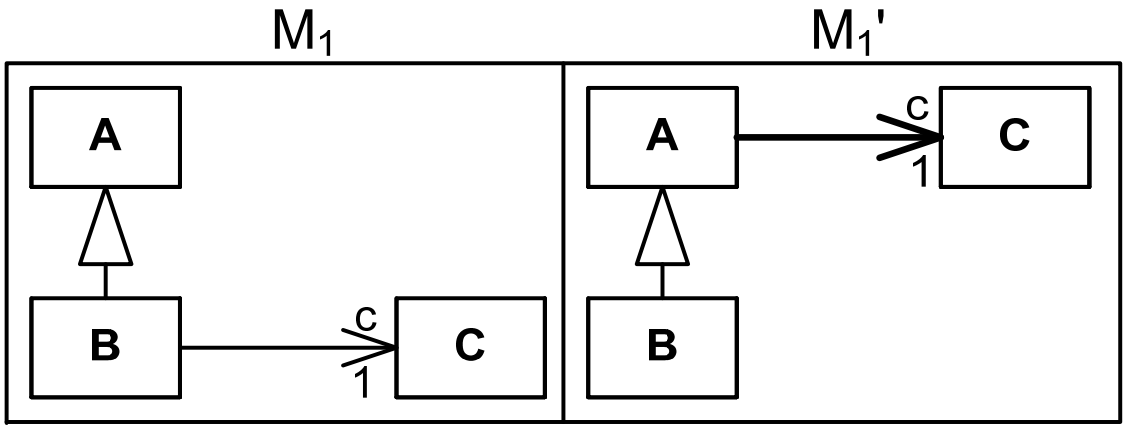
Monitoring could be enhanced if conceptual models such as requirement and system models would be considered and related at runtime:

How to facilitate
root cause analysis of runtime violations
at the abstraction level of models?

Overview of Contributions



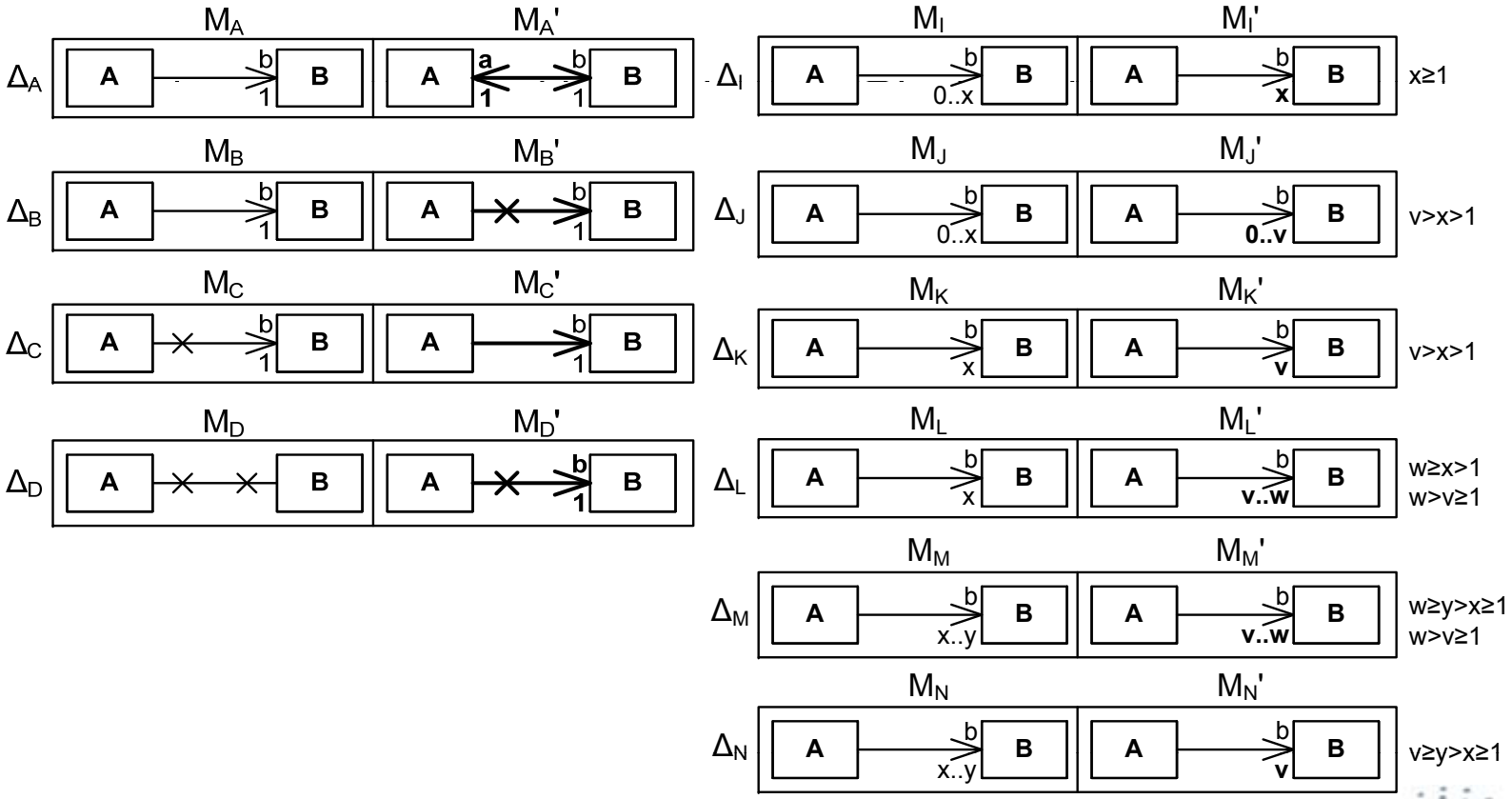
Navigation Compatible Changes



Definition

Let f be a bijective mapping function M → M' between model elements m ∈ M and m' ∈ M' that have the same model element type.

A model M' is navigation compatible to a model M if ∀ concrete classes c ∈ M there ∃ a class f(c) ∈ M' and if ∀ references r ∈ c referencing a class d ∈ M there ∃ a reference f(r) ∈ f(c) that references a class or subclass of f(d) ∈ M'.

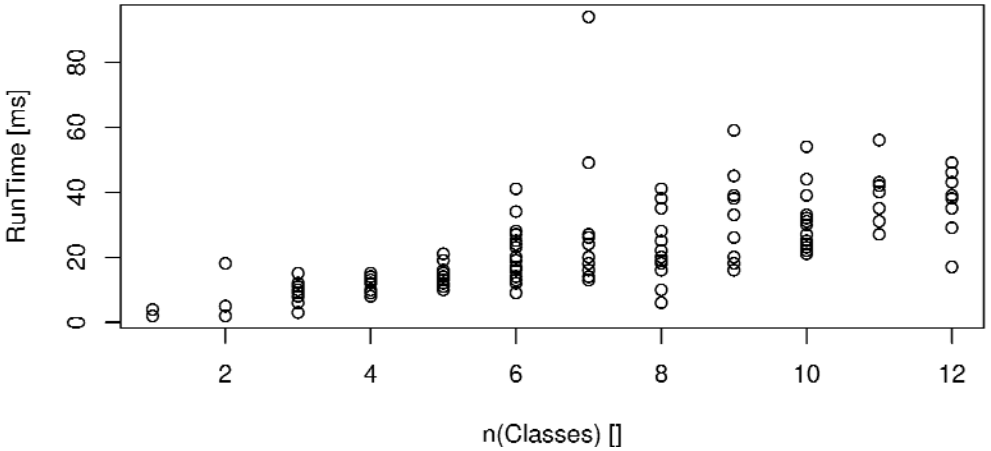


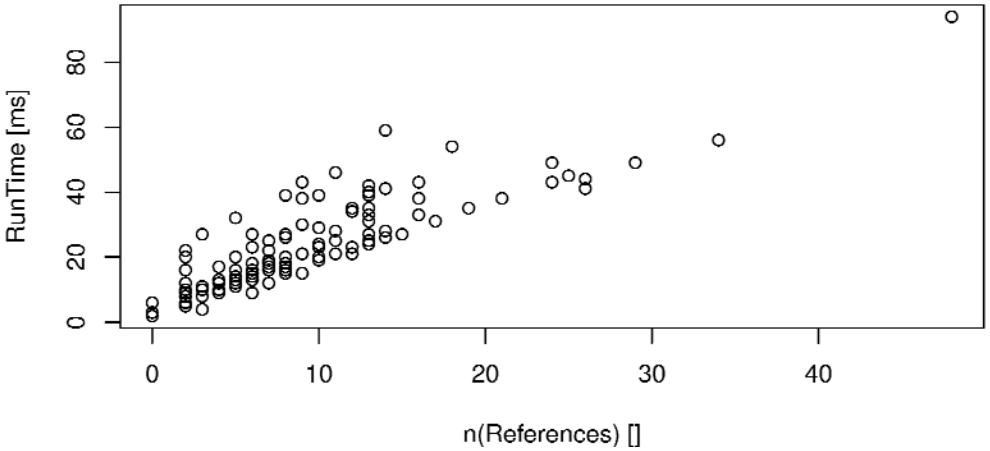
Navigation Compatibility Check

pp.22-28

Real World Meta-Models

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	
1.0	6.0	12.0	19.5	26.0	114.0	n(Classes)
0.00	2.00	7.50	14.42	16.25	96.00	n(Generalizations)
0.00	6.00	13.00	24.08	27.25	231.00	n(References)
0.0000	0.2857	0.6000	0.5537	0.8632	1.7140	n(Generalizations) / n(Classes)
0.0000	0.7411	1.1500	1.2450	1.5770	6.8570	n(References) / n(Classes)
2.0	**18.0**	**39.5**	**87.1**	**98.5**	**725.0**	**RunTime [ms]**





Applied in an Industrial Setting

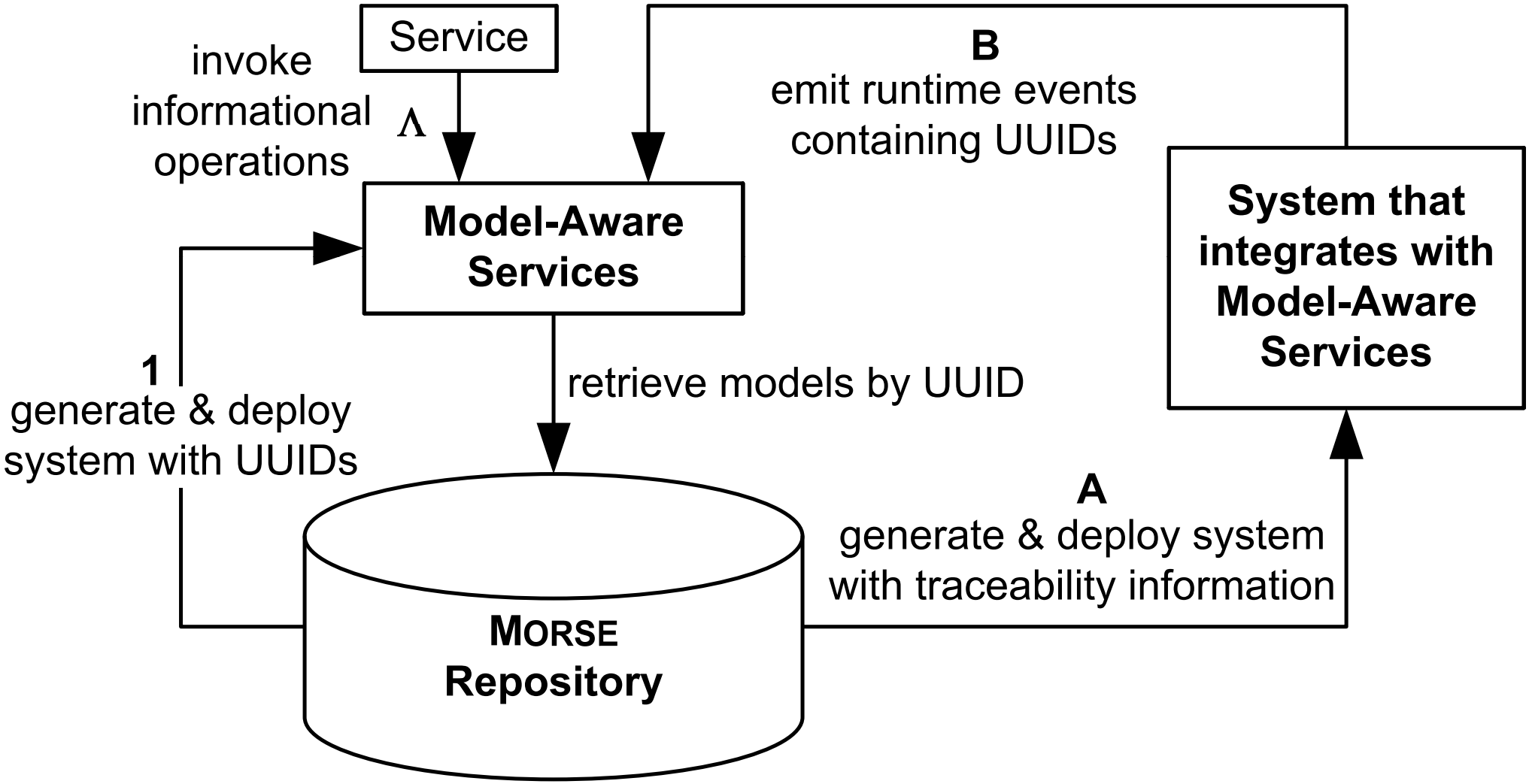
during the development and evolution

- ensuring the downwards-compatibility of evolved meta-model versions
- MDE developers are free to undertake navigation compatible changes frequently
- changes that break navigation compatibility require a formal agreement of the developers as this involves the co-evolution of other artifacts

➢ ***Lines** of Navigation Compatibility*

➢ ***Moments** of Navigation Incompatible Changes*

The Model-Aware Service Environment (MORSE) Approach



Model-Aware Systems

contain, emit, or use model traceability information for model look-up and reflection

Model-Aware Services

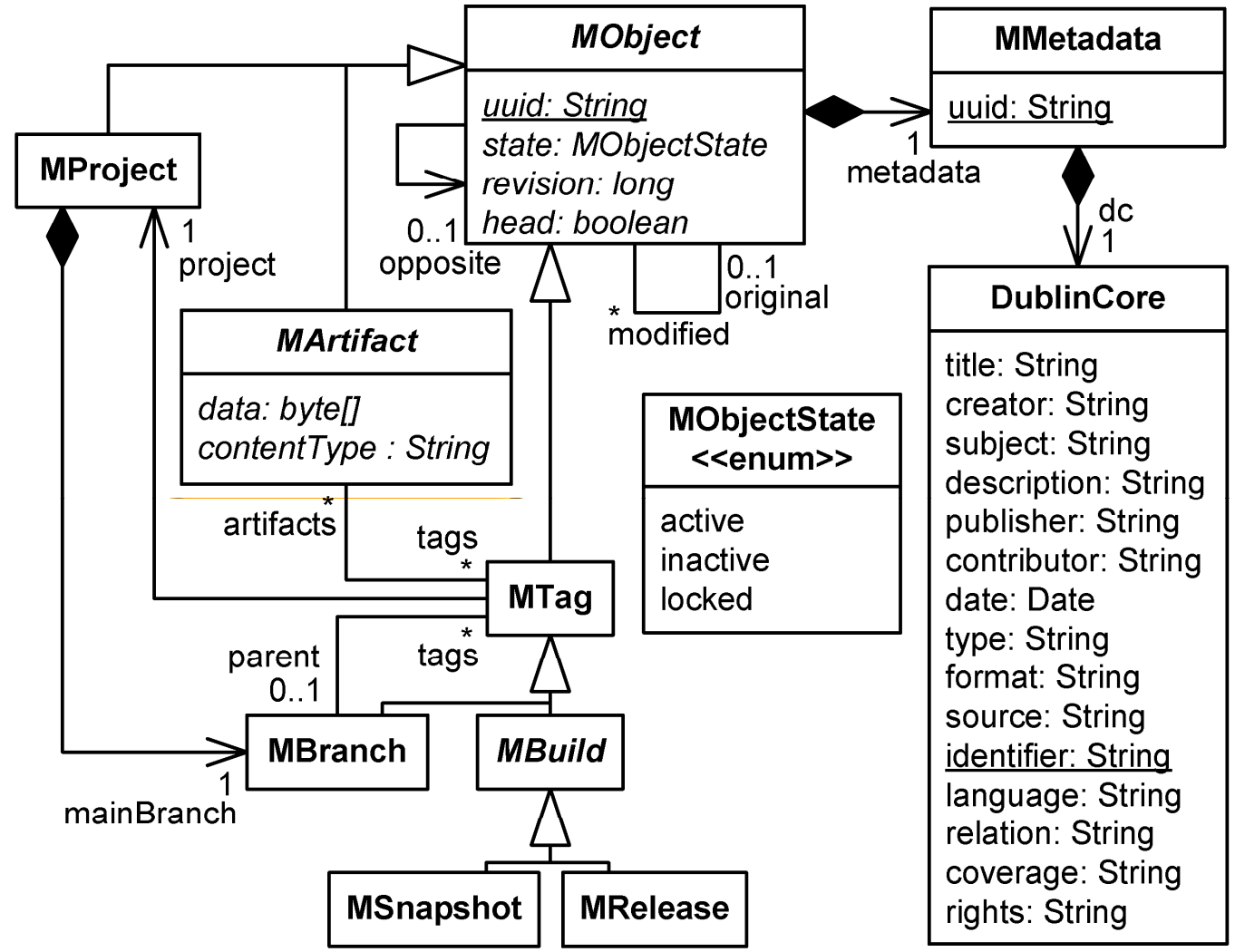
- UUIDs are embedded during transformation
- may interact with MORSE (e.g., retrieve models from which they have been generated)
- may expose traceability information

Model-Aware Processes

- emit UUIDs during process execution

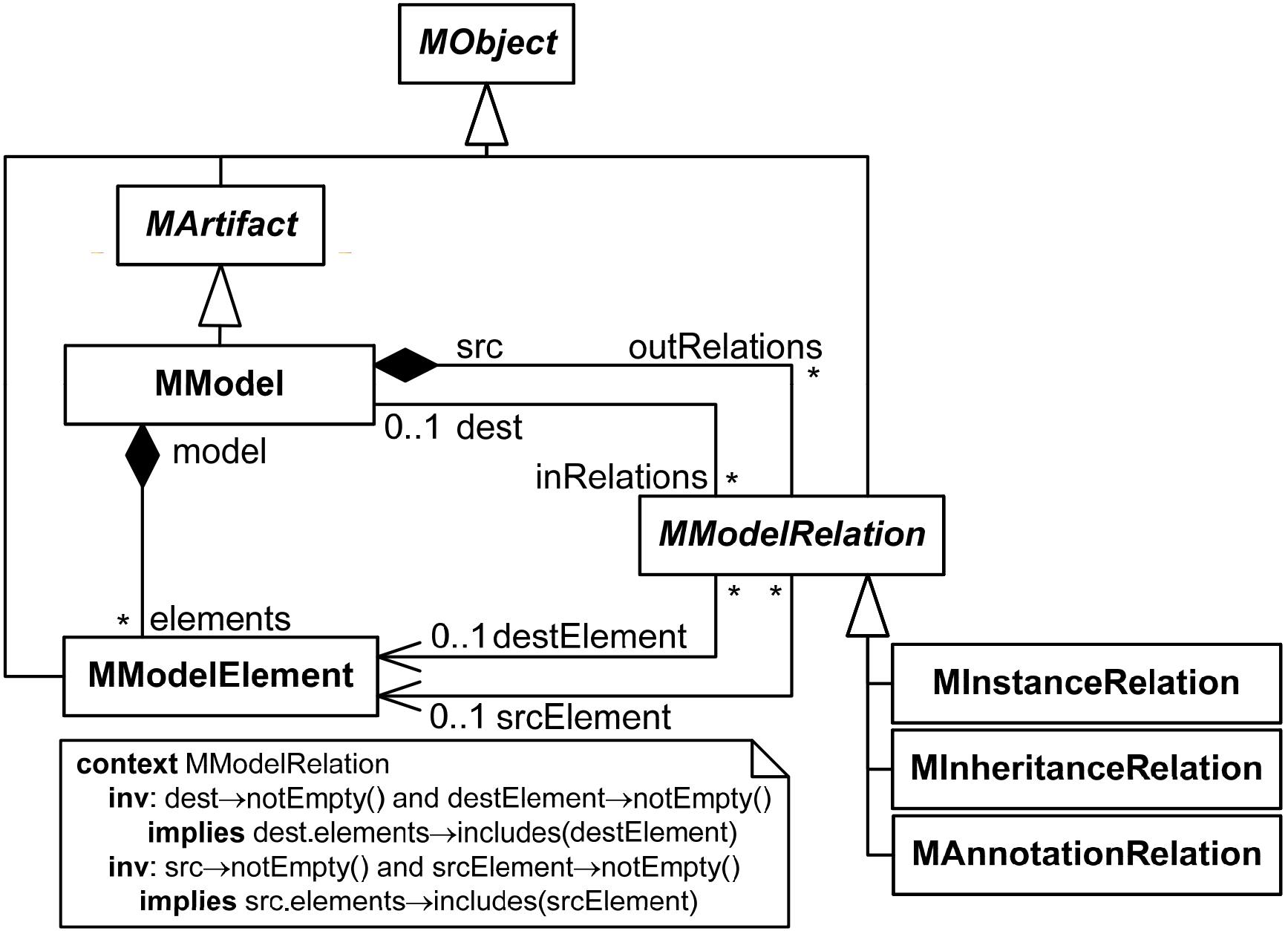
  proposed and realized as a BPEL-extension

MORSE Core Model



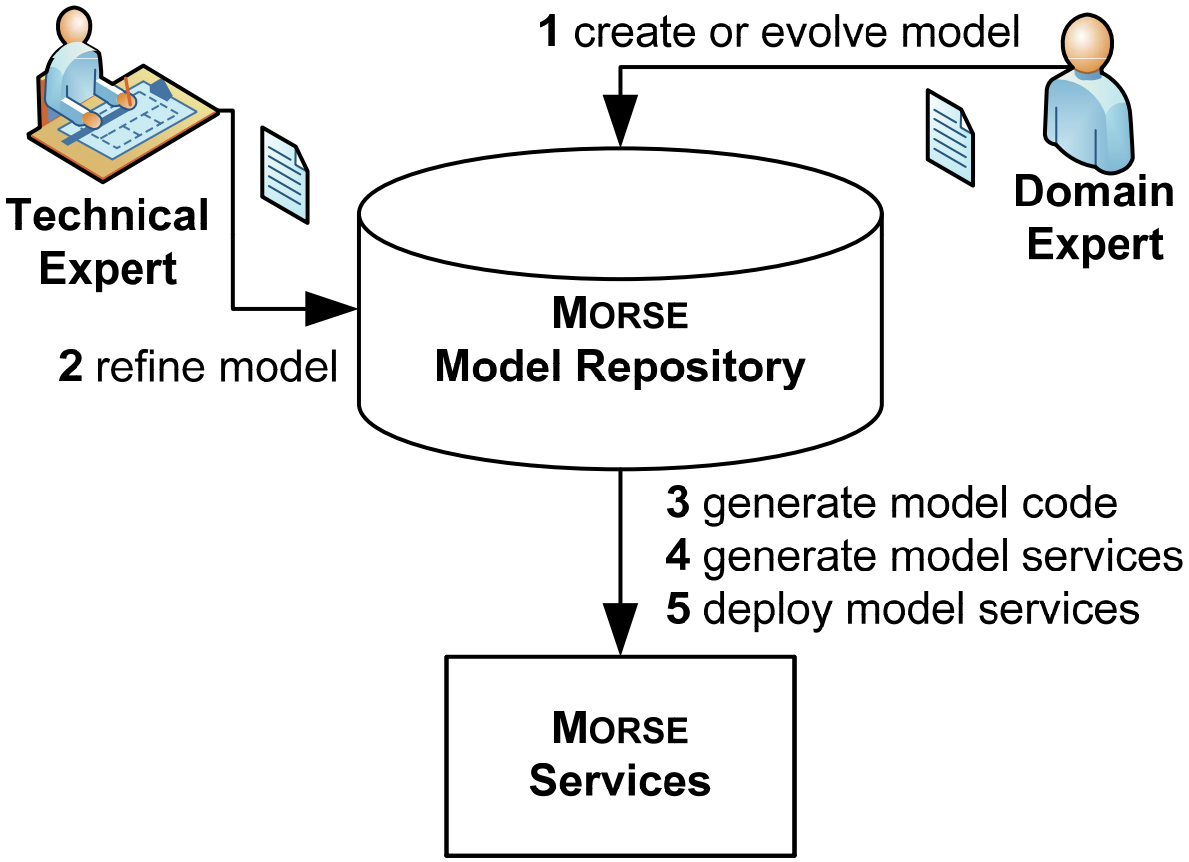

```
context MObject
    inv: metadata.uuid = uuid and metadata.dc.identifier = uuid
    inv: opposite→notEmpty() implies ((head xor opposite.head) and opposite.oclIsTypeOf(self))
    inv: original→notEmpty() implies original.oclIsTypeOf(self)
context MProject inv: mainBranch.project = self
context MTag inv: parent→notEmpty() implies parent.project = project
```

MModel, -Element, -Relation



MORSE Services

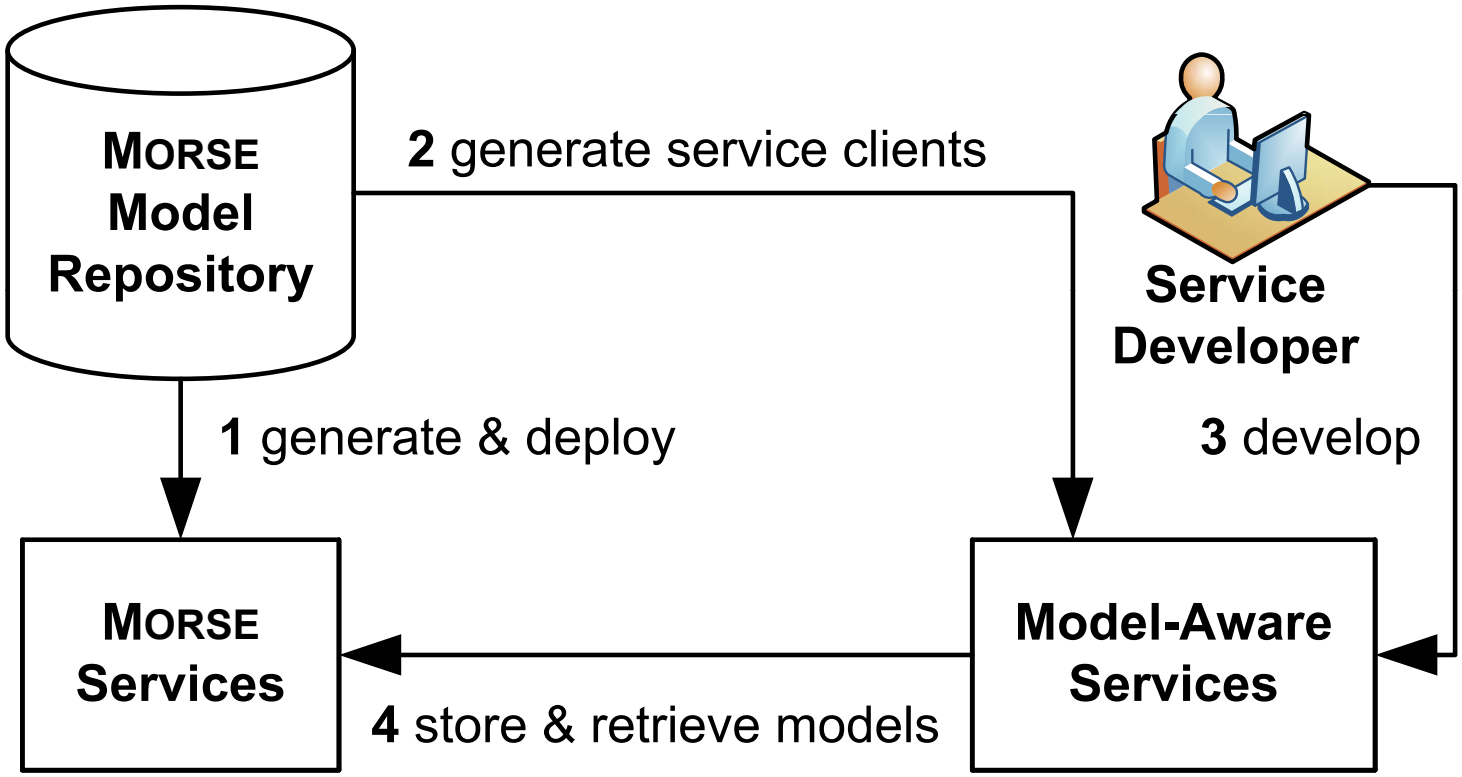
- generated for each concept of a model
  - information retrieval
  - resource management
  - versioning



MORSE Service Operations

Response	Operation	Description
boolean	**exists**	does a model with a UUID exist?
boolean	**isHead**	is the object (specified by UUID) version-independent?
UUID[]	**list**	returns the VIIDs of all models
UUID[]	**versions**	returns all VSIDs of a model
<Class>[]	**query**	search for models; support of various query parameters
<Class>	**retrieve**	a model is retrieved by UUID
UUID	**create**	a VIID is returned
UUID	**update**	a VSID is returned
UUID	**delete**	a VSID is returned
UUID[]	**list<Role>**	returns the UUIDs for a role
UUID	**add<Role>**	a VSID is returned
UUID	**remove<Role>**	a VSID is returned
UUID	**clear<Role>**	a VSID is returned

MORSE Service Clients



Model Repository Comparison

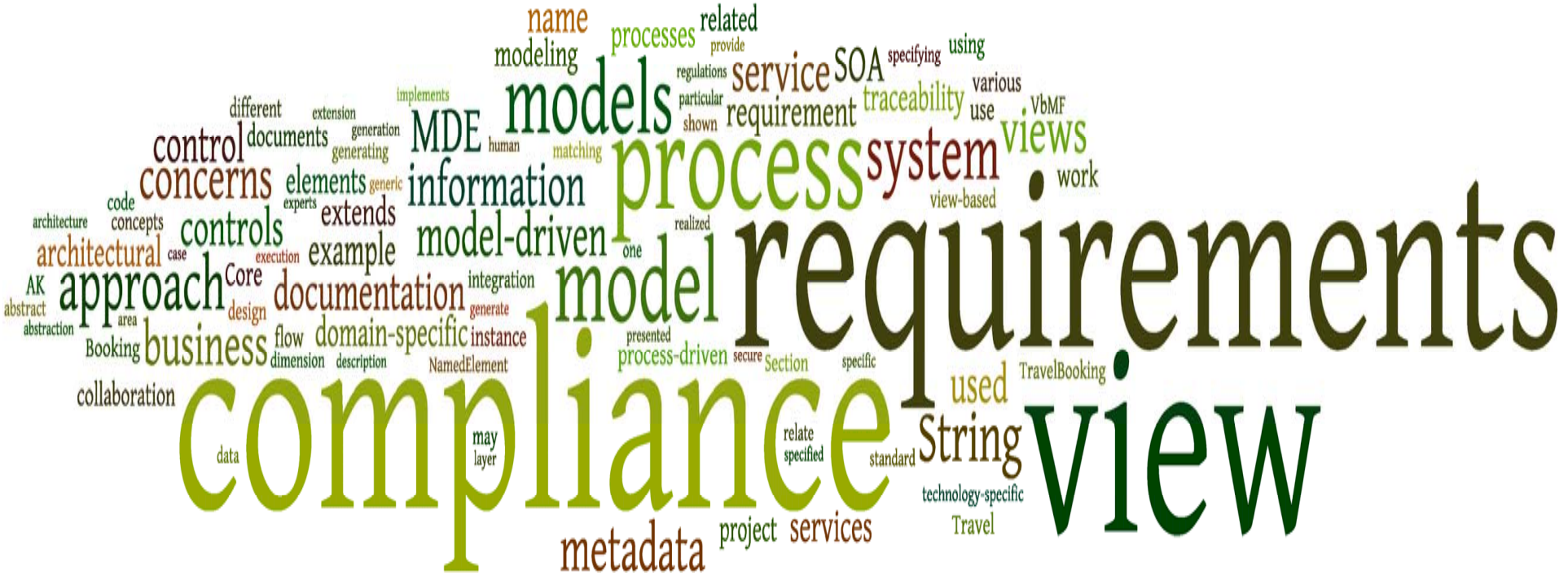
pp.50,52

Repository	Model Identification	Model Element Identification	Model Navigation	Complex Search
AMOR	URL	ID	✗	✗
AtlanticZoo	URL	✗	✗	✗
CDO	URL	URI-Fragment	✓	✓
EMFStore	ID	ID	✓	✗
MDR	ID	URI-Fragment	✗	✗
ModelBus	URL	✗	✗	✗
MORSE	**UUID**	**UUID**	✓	✓
Odyssey-SCM	ID	URI-Fragment	✗	✗
Odyssey-VCS 2	ID	URI-Fragment	✗	✗

CHAPTER 5
Linking Systems and Requirements through Model-Driven Engineering

A good marriage is one which allows for change and growth.

— PEARL BUCK [1]



[1] 賽珍珠

A Practical Modeling Approach

pp.69-83

Focus: Business Process Compliance

Models are used for describing the

- System

  Process View Models

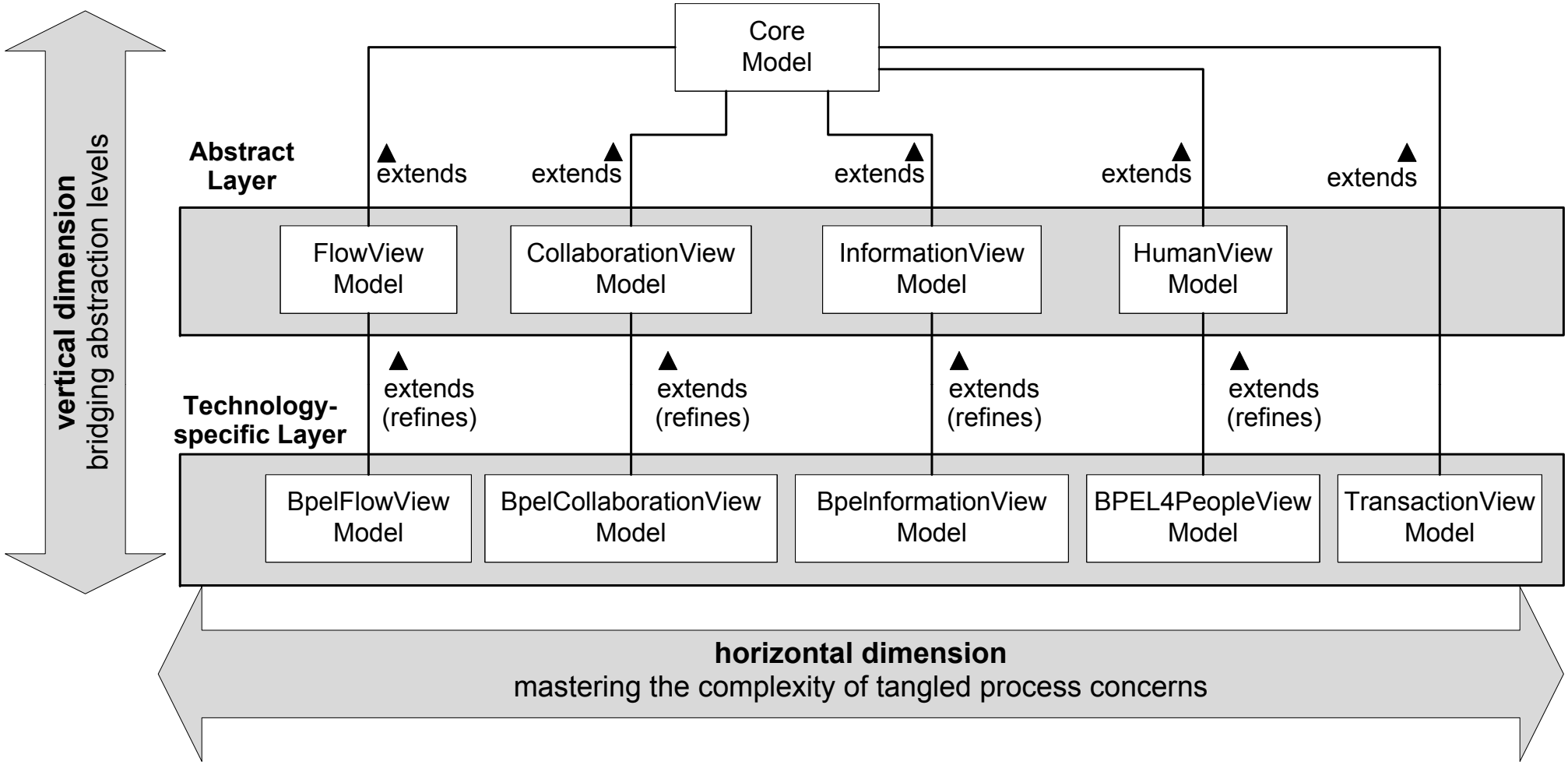
- System Requirements

  Compliance Concerns of Business Processes

Requirement Models **annotate** System Models

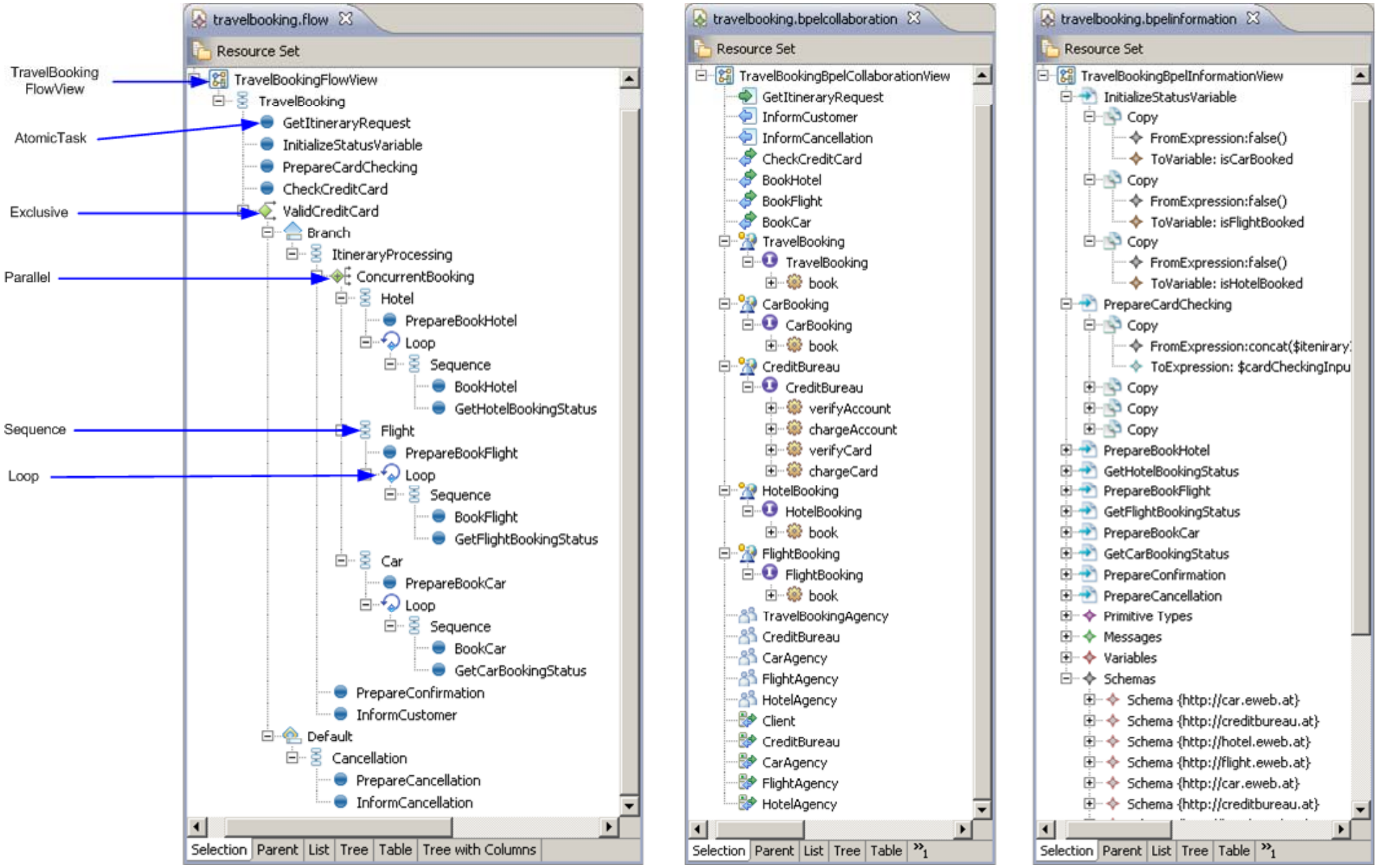
⇒ *novel, direct linkage and correlation of system & requirements models*

View-based Modeling Framework

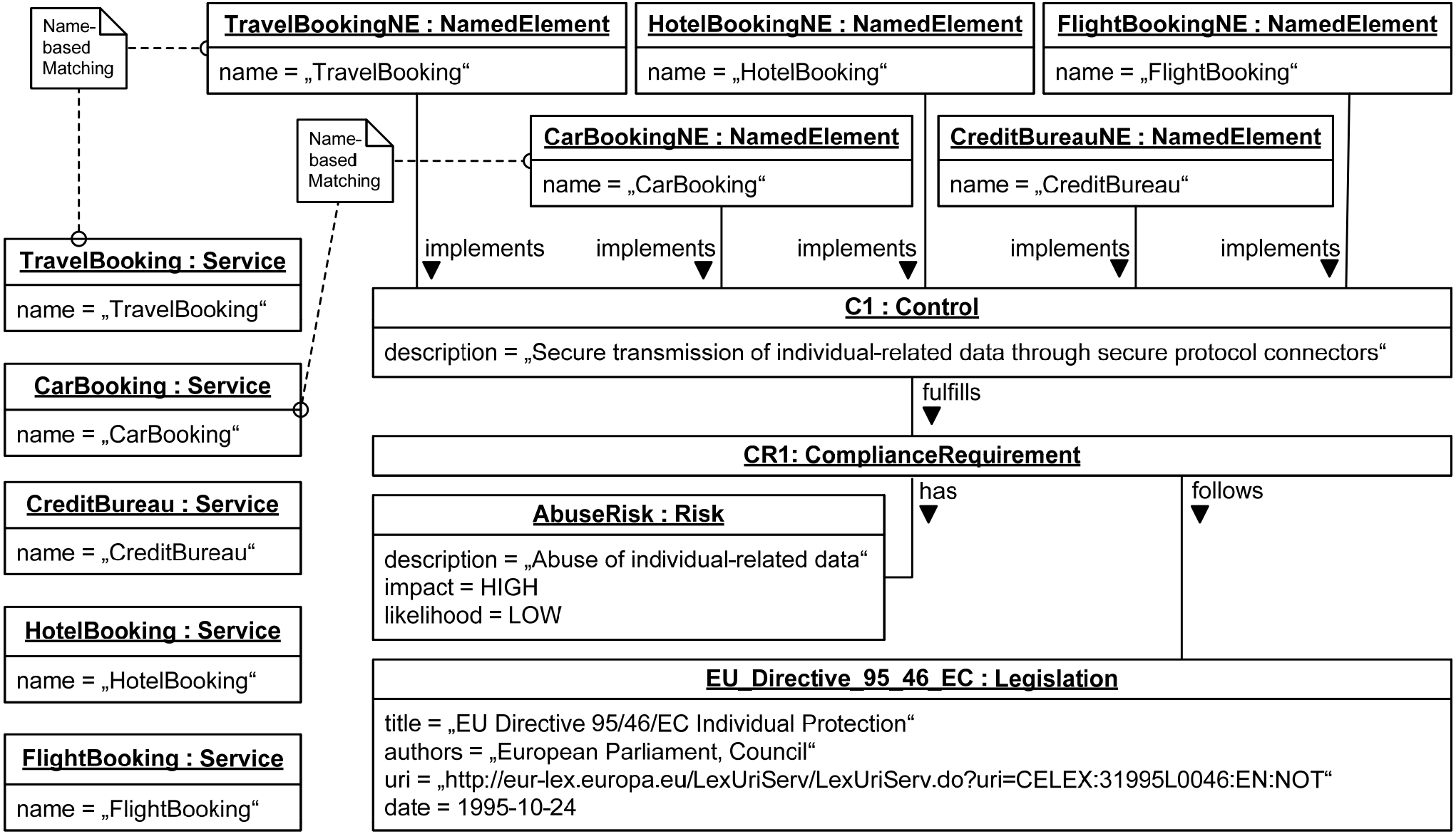


Huy Tran, Uwe Zdun, and Schahram Dustdar.
"View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA".
In: BPSC. Vol. 116. LNI. GI, 2007, pp. 105–124.

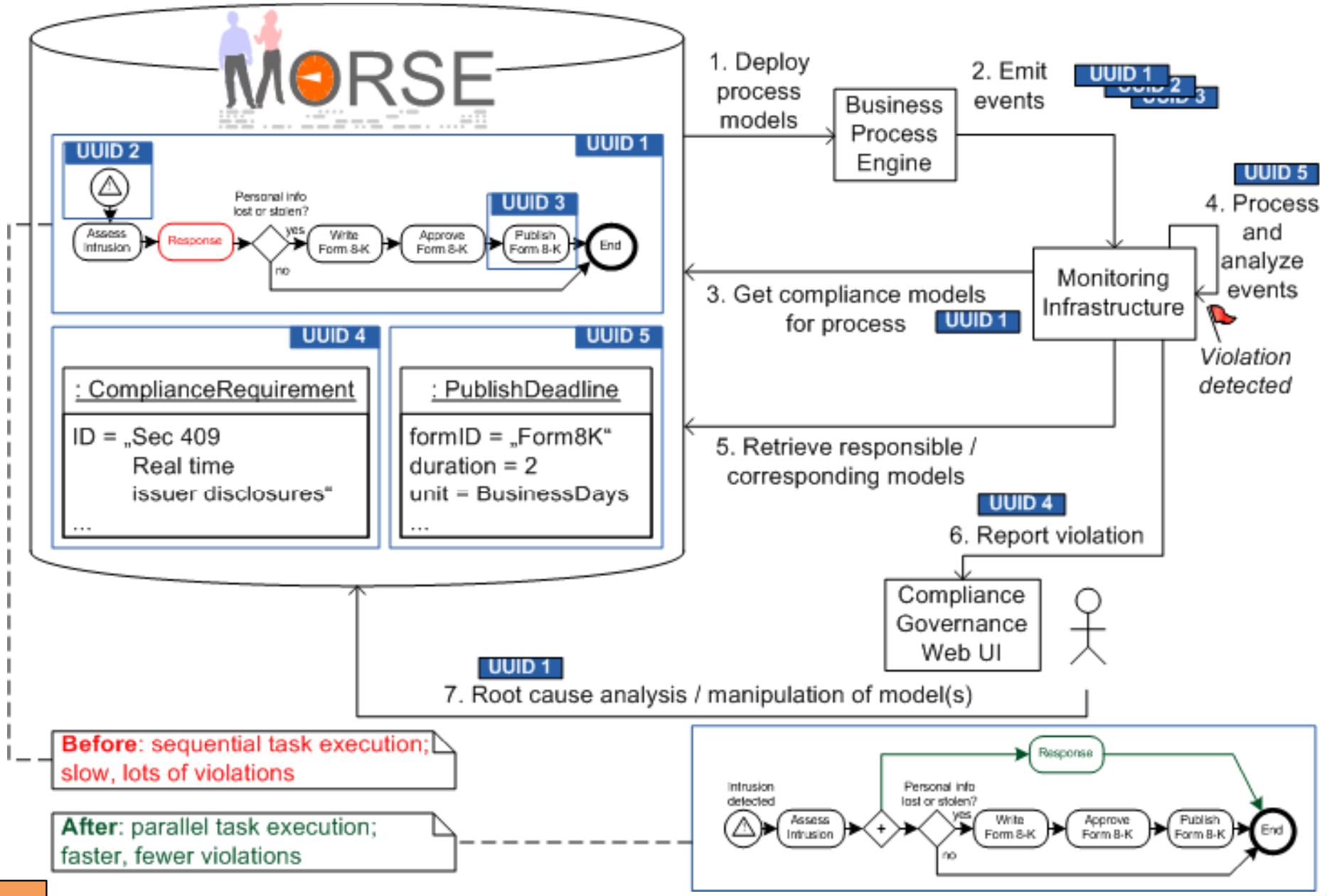
System Models: Process Views



Requirement Model: Compliance Metadata Instance



Model-Aware Monitoring



Main Scientific Contributions

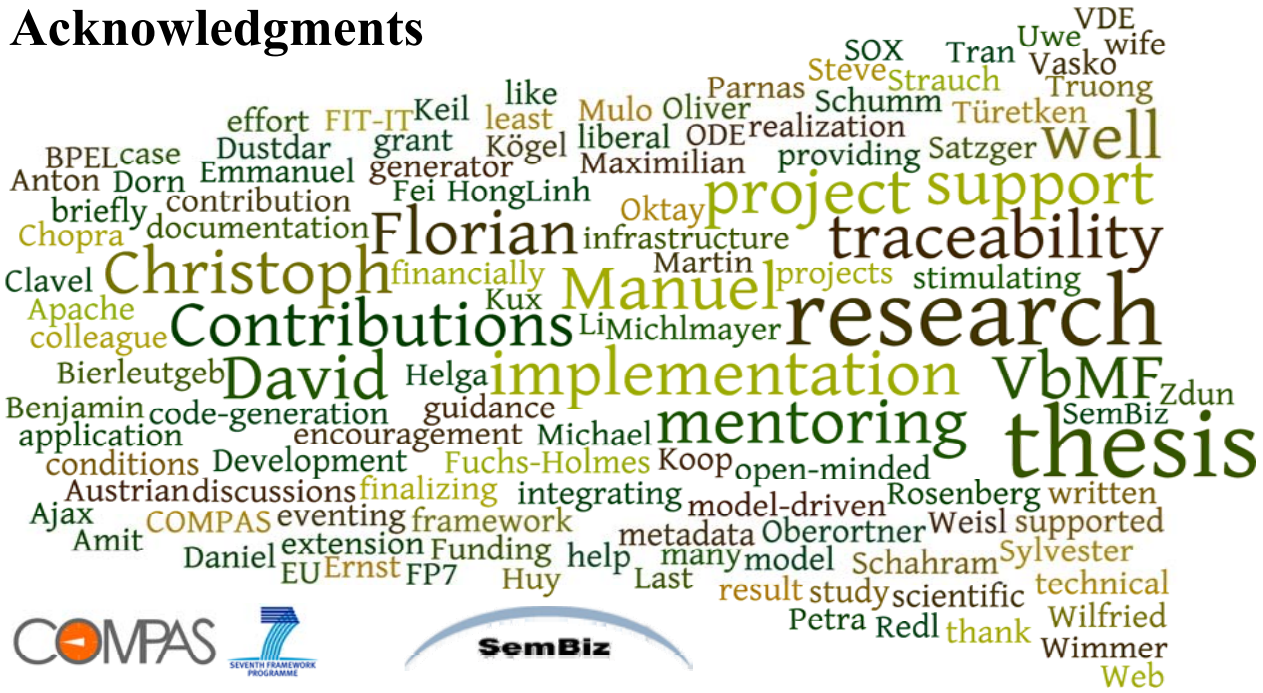
- We defined the *novel* notion of a navigation compatible model change and presented an algorithm for determining navigation compatibility.
- In a SOA we made MDE artifacts uniquely identifiable and retrievable and facilitated their design- and runtime use and management.
- We demonstrated a direct linkage of system & requirement models, presented model-aware systems and realized model-aware monitoring.

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Thanks for your participation and attention!

Ta‘id HOLMES

Acknowledgments



VITALAB
VIENNA INTERNET TECHNOLOGIES ADVANCED RESEARCH LAB
DISTRIBUTED SYSTEMS GROUP