42. Context-Role-Based Architecture of MDSD Tools and Materials in a Technical Space (on M0)

Prof. Dr. Uwe Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de/teaching/most

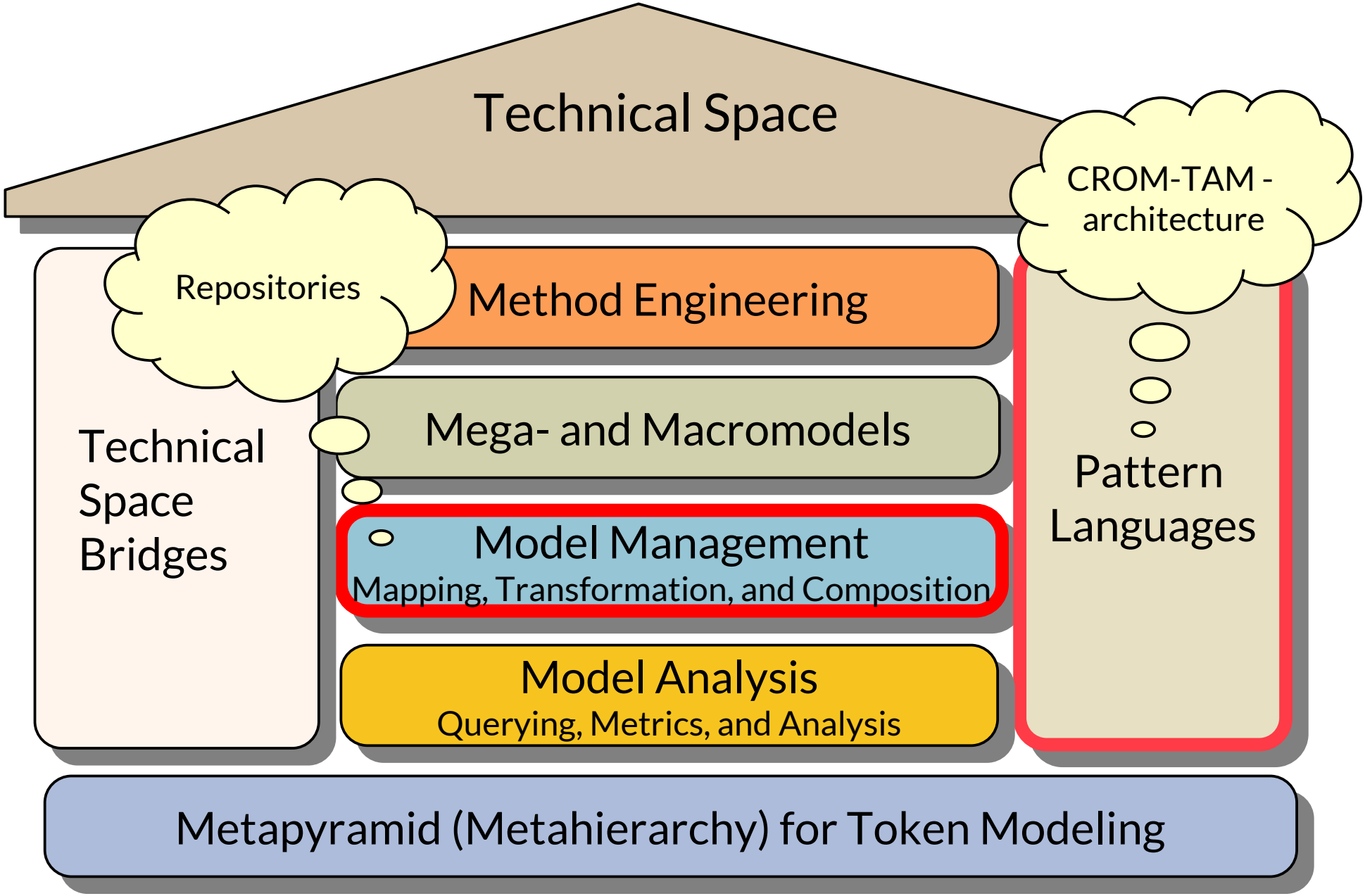
Version 17-1.1, 15.01.18

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Obligatory Literature

- Netbeans Metadata repository (MDR) http://docs.huihoo.com/netbeans/netbeanstp.pdf
- [deep-ROP] D. Bäumer, D. Riehle, W. Silberski, M. Wulf. Role Object. Conf. On Pattern Languages of Programming (PLOP) 97. http://citeseer.ist.pst.edu/baumer97role.html
- Steffen Staab, Tobias Walter, Gerd Gröner, and Fernando Silva Parreiras. Model driven engineering with ontology technologies. In Uwe Aßmann, Andreas Bartho, and Christian Wende, editors, Reasoning Web, volume 6325, Lecture Notes in Computer Science, pages 62-98. Springer, 2010.
  - http://www.uni-koblenz.de/~staab/Research/Publications/2010/reasoningweb2010.pdf

Q10: The House of a Technical Space



Importance of CROM for Metamodel-Driven Repositories

- Concptually, tools and materials collaborate with collaborations (roles in contexts/compartments)
- CROM is an ideal modeling language for tool architecture
  - and is used in the following.

42.1 Context-Role Based Architecture of Tools in TAM
(CROM-based architecture of TAM)

Tools in an IDE (on M0) are built with the pattern language Tools- Automata – Material (TAM) using contexts and roles

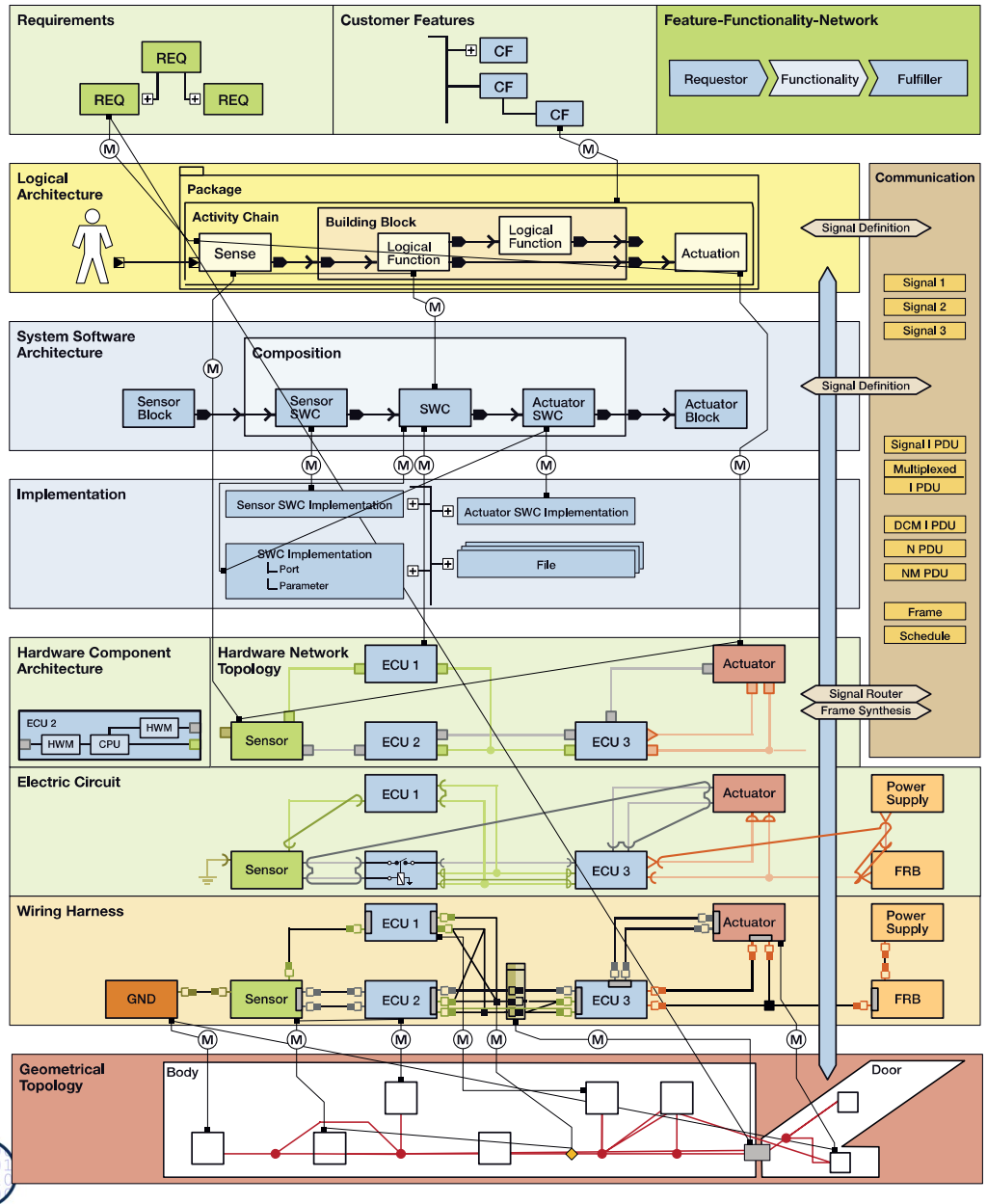
Cmp. The architectural styles of applications from Softwaretechnologie-II:

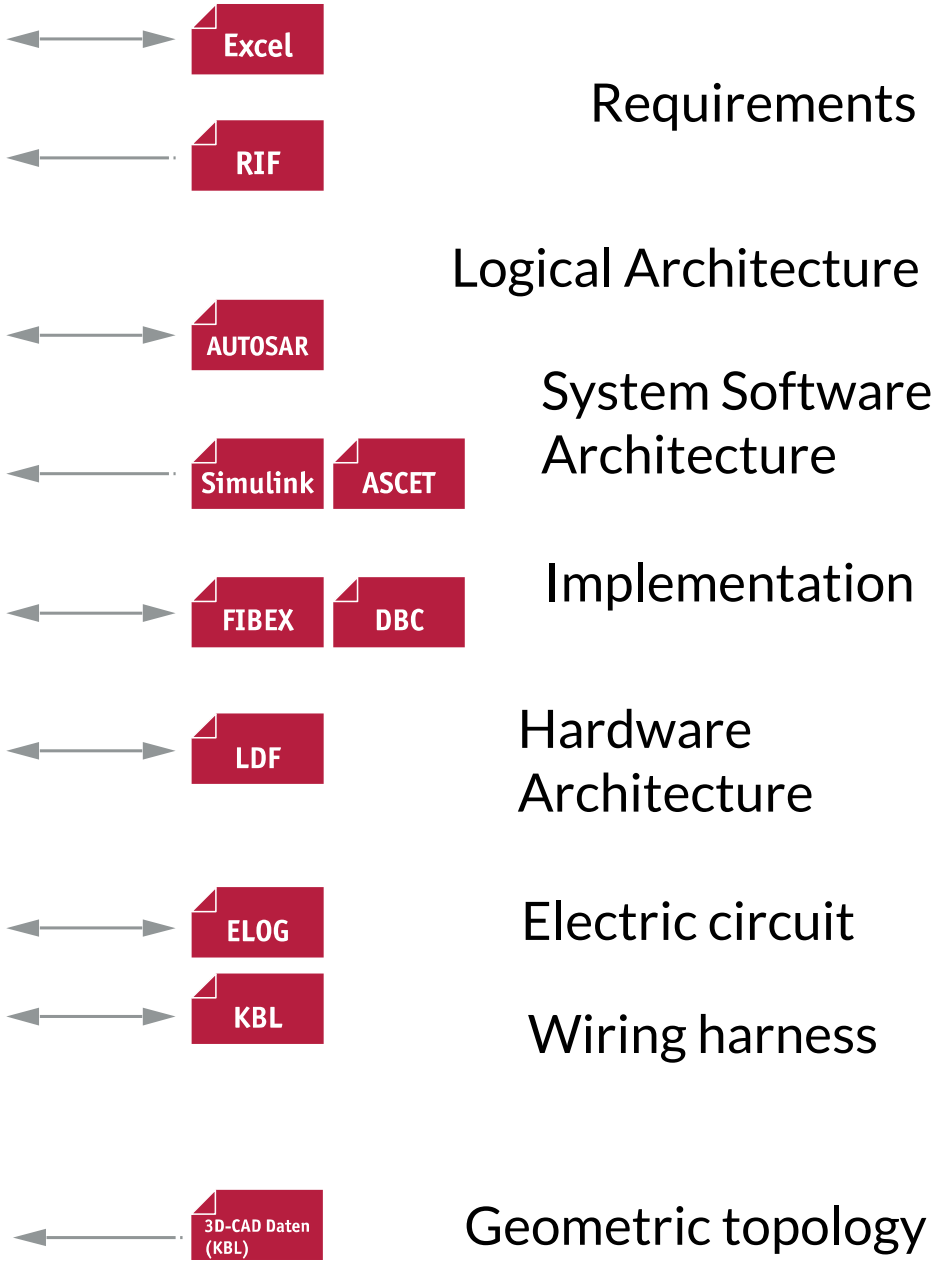
Algebraic Applications: based on calls

Coalgebraic Applications: based on streams

Motivation: Coarse-Grain Picture: Tools and Materials in the Car Design with PREEVision (Vector)

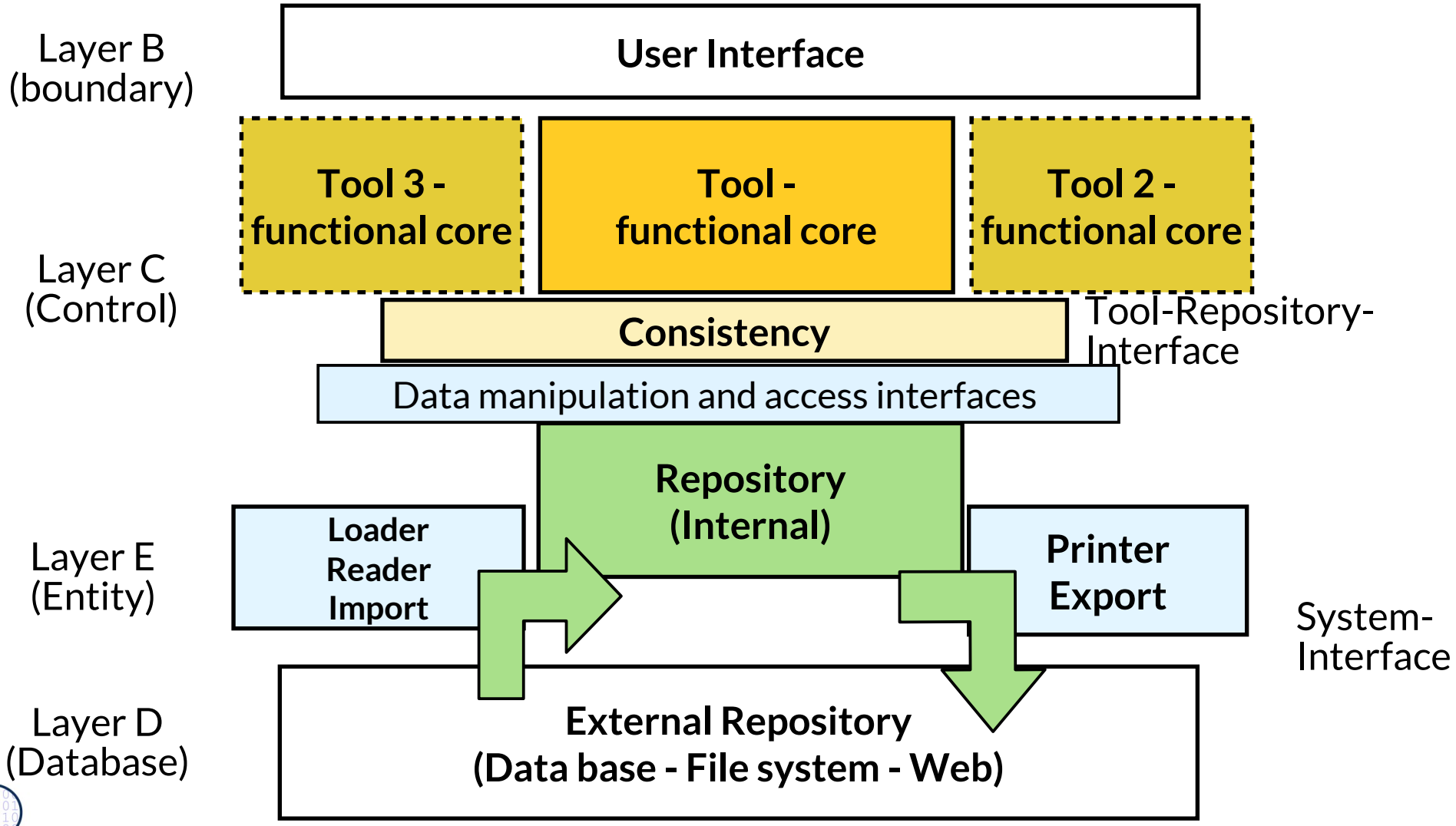
[Preevision]





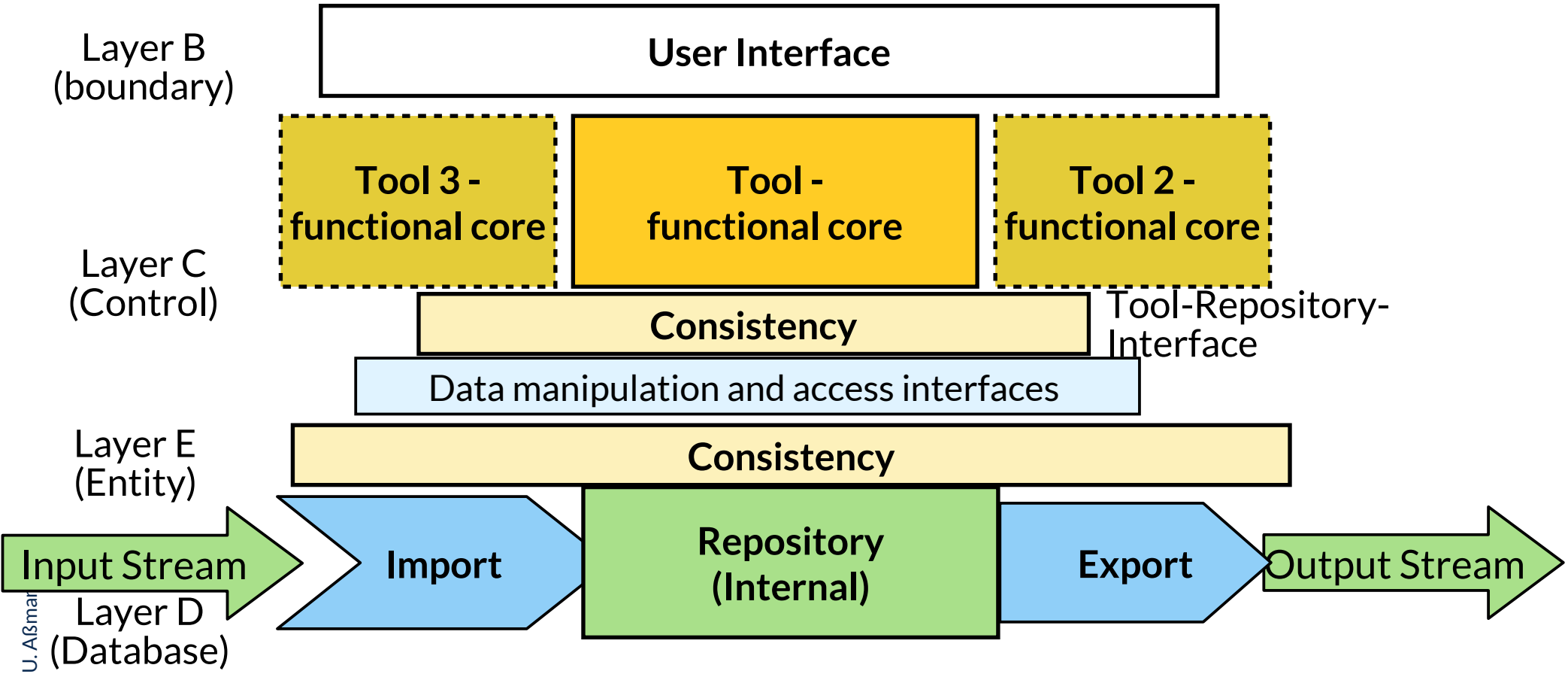
Q5: Generic Architecture of Repository-Based Tools (call-based Repository)

- Repository may be shared between several tools



Q6: Architecture of Stream-BasedTools

- Usually, the repository is internal and non-shared
- Therefore, the work is done piece by piece, as it arrives via the stream



42.2. Metamodel-Driven Repository for the Materials

- A **Repository (Datenbasis, Datenablage)** is the container of all Materials of a software system.
- A Repository stores
  - Material objects from M0
  - Models from M1
  - Metamodels from M2

Objectives of a Repository

- Cooperation of Tools
  - **Synchronisation** on metadata and Materials
  - **Transactions** for consistent change of Materials
  - **Configuration management**: Varianting and versioning
- Collaboration:
  - **Transparent Distribution:** access from everywhere in the company or internet
  - **Efficient access**: Transactions per second
  - Event-based communication: Notification when changes occured
- **(Transparent) Persistent storage** of Materials from M0, M1, M2
  - Physical data model: should be transparent to the application
    - often in entity-relational schema, then management of an object-relational mapping
- **Administration**: management interfaces
- Development qualities:
  - **Adaptability**: Embedding of repository into applications, IDE and MDSD tools
  - **Extensibility**
  - **Portability**

Quelle: nach Habermann, H.-J., Leymann, F.: Repository - eine Einführung; Oldenbourg Verlag 1993
und Däberitz, D.: Der Bau of SEU mit NDBMS; Diss. Uni Siegen 1997

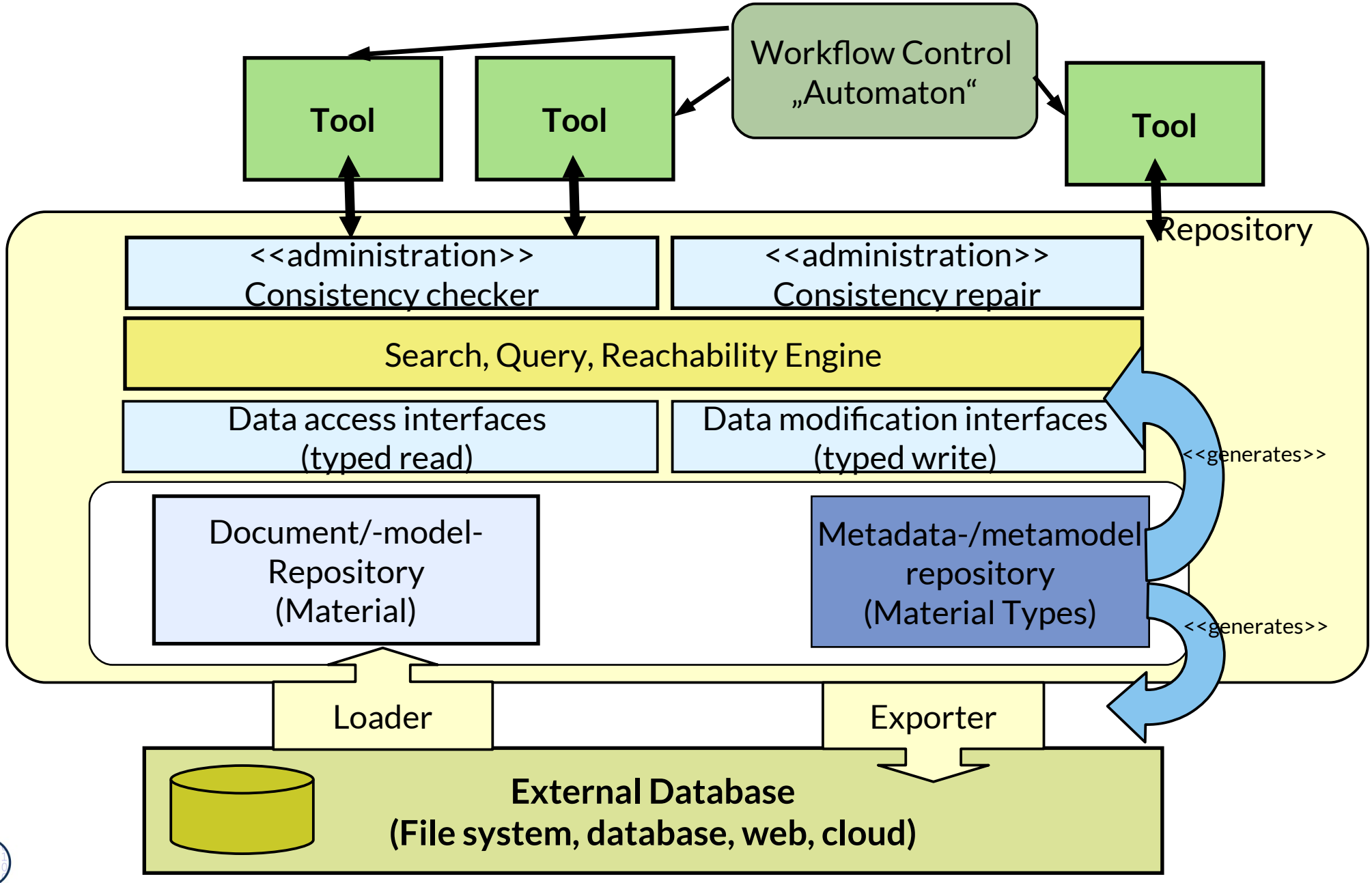
42.2.1 Metamodel-Driven Repositories

Objectives of a Repository

- Support by **metamodels** on M2, of DDL describing the structure of Materials:
  - **Typed access interfaces**: programmable APIs
  - **Untyped access (reflective) interfaces**
  - **Query languages on models and objects**
  - **Logical data model:** application-specific data model
  - **External metamodels** for import and export
    - Textual
    - Exchange formats, such as XML or JSON

Quelle: nach Habermann, H.-J., Leymann, F.: Repository - eine Einführung; Oldenbourg Verlag 1993 und Däberitz, D.: Der Bau of SEU mit NDBMS; Diss. Uni Siegen 1997

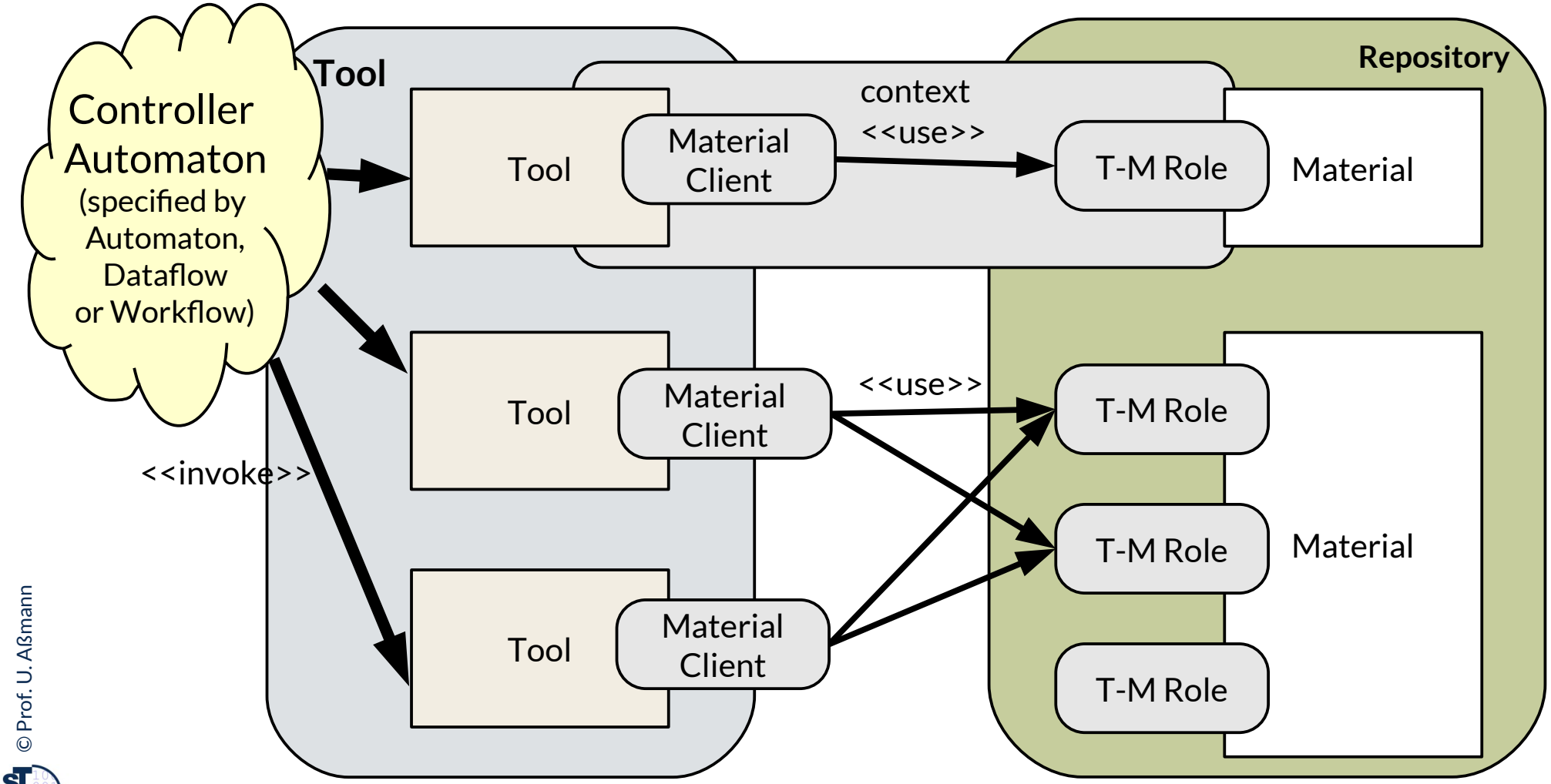
Q7: Tool Architecture with Data Sharing in a Metamodel-Driven Repository



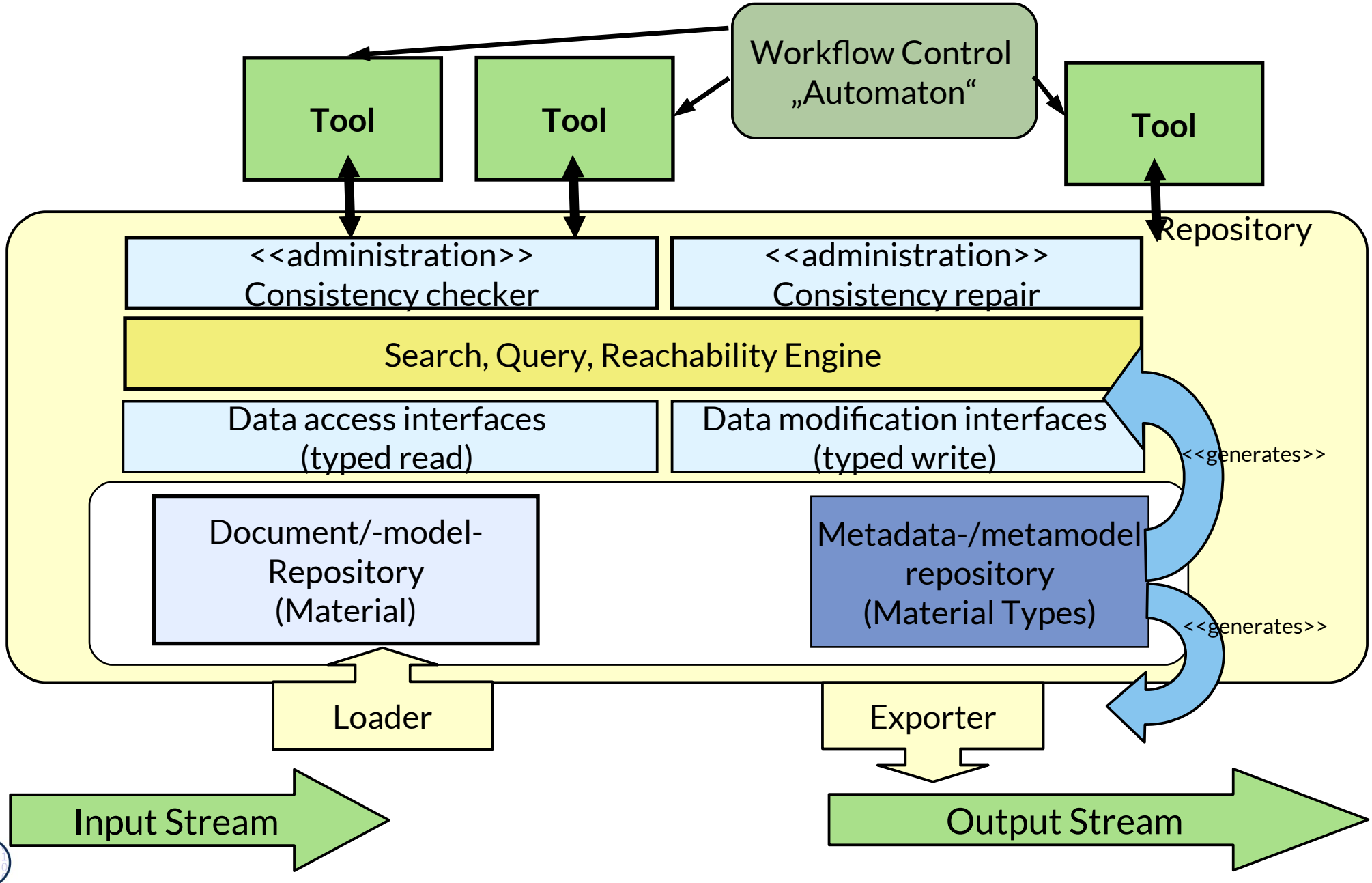
Controller Automaton, Tools and Their Views on Material

- Tools are controled by automata or workflows (TA
- Sometimes, we omit the context (if it is clear)

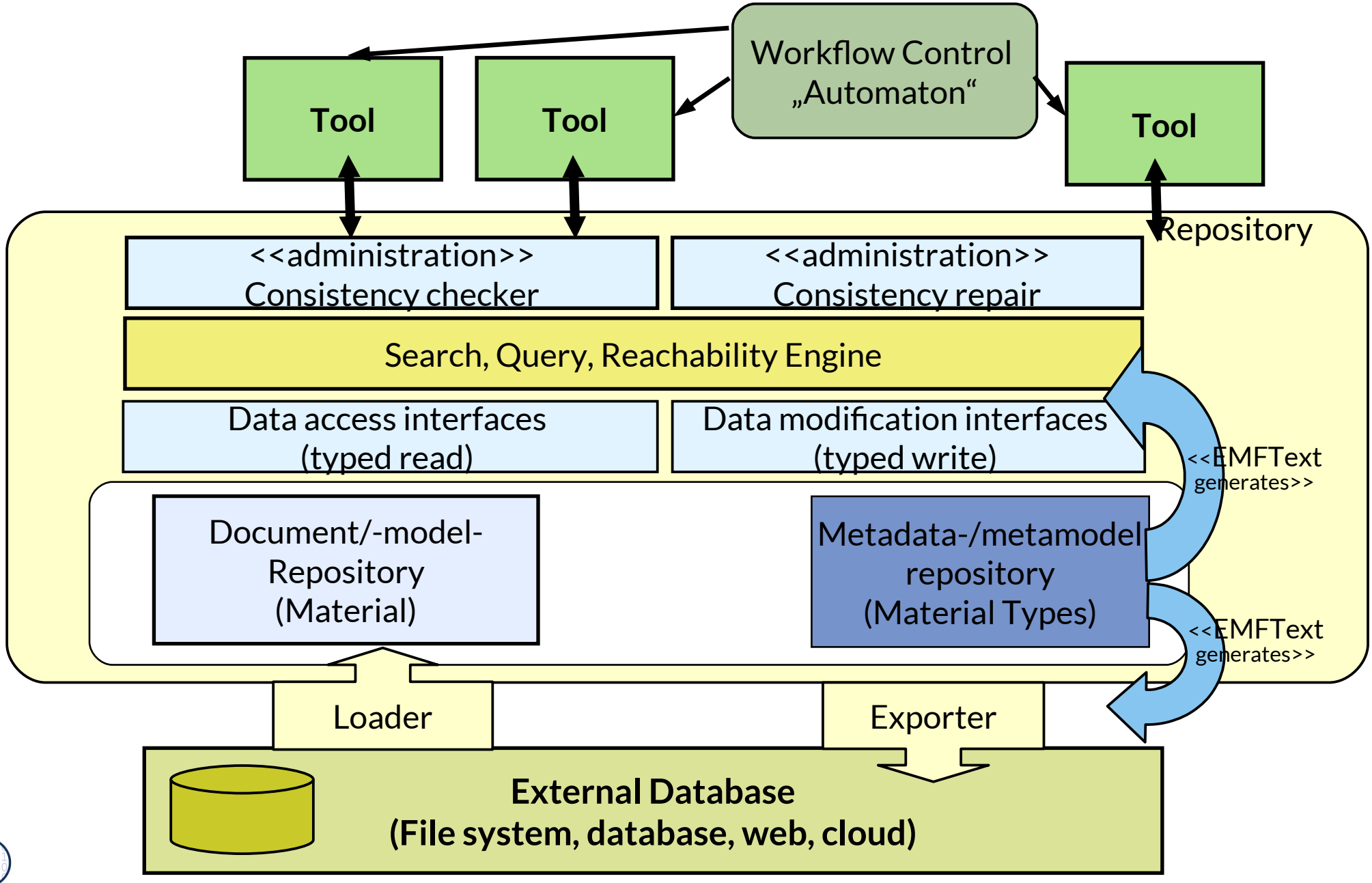
Tafelbild



Q7.b: Stream-Based Tool Architecture with a Metamodel-Driven Repository



Tool Architecture with Data Sharing in EMFText-Driven Repository

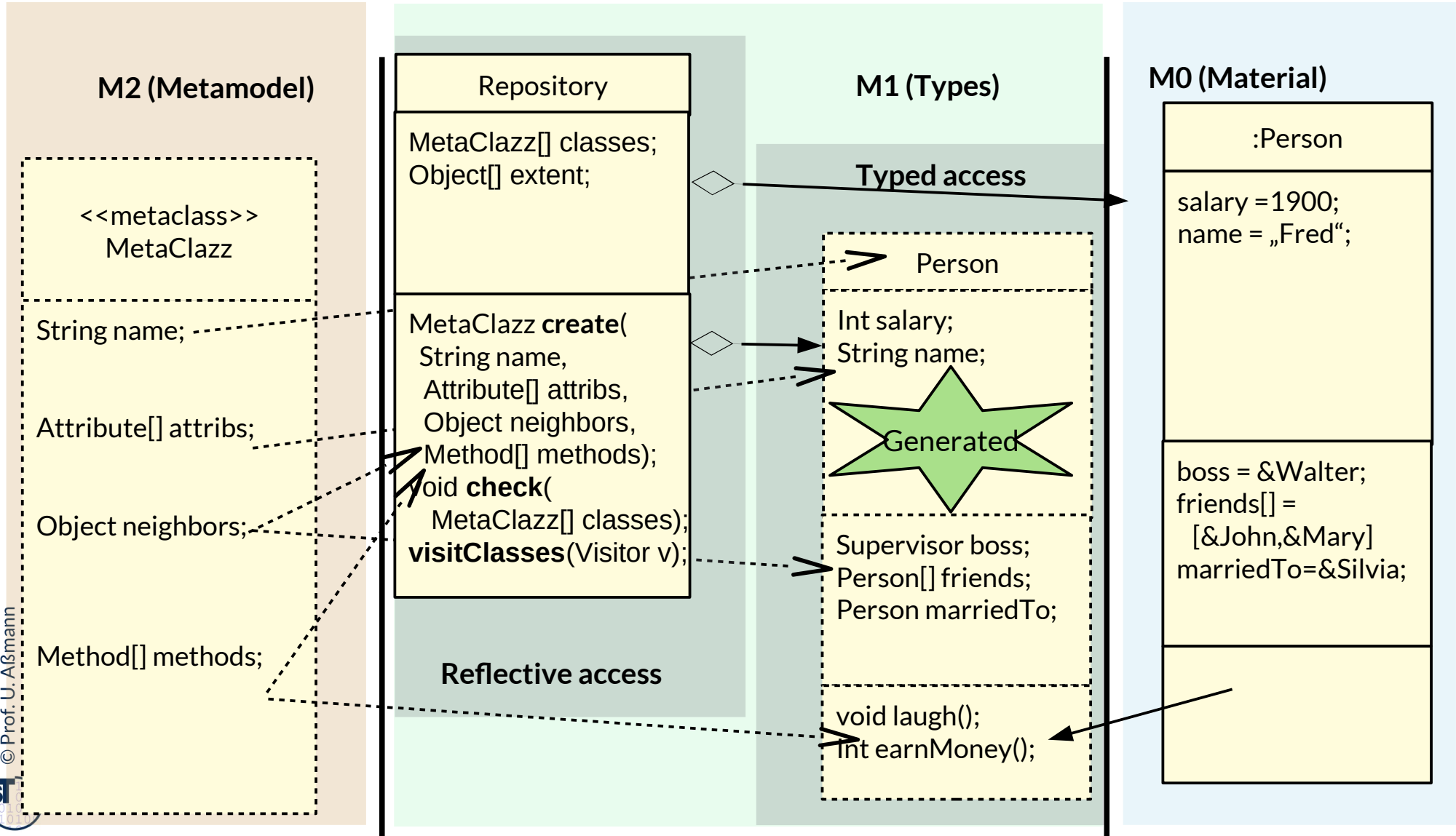


What are the Benefits of a Metamodel-Driven Repository?

- A **metamodel-driven repository** enables to create typed repositories with typed Materials
  - Load one or several metamodels into the metadata (sub-)repository
  - Metamodels must be defined in the same metalanguage (or lifted DDL)
- Benefits:
  - Structural information about the objects and models conforming to the metamodels is known
    - Members of classes and objects
    - Atomic, set-based, structured or reference attributes
    - Collections or graphs
    - Cardinalities of neighbors
  - From these informations typed access interfaces to the Materials can be generated

Generation of Typed Interfaces

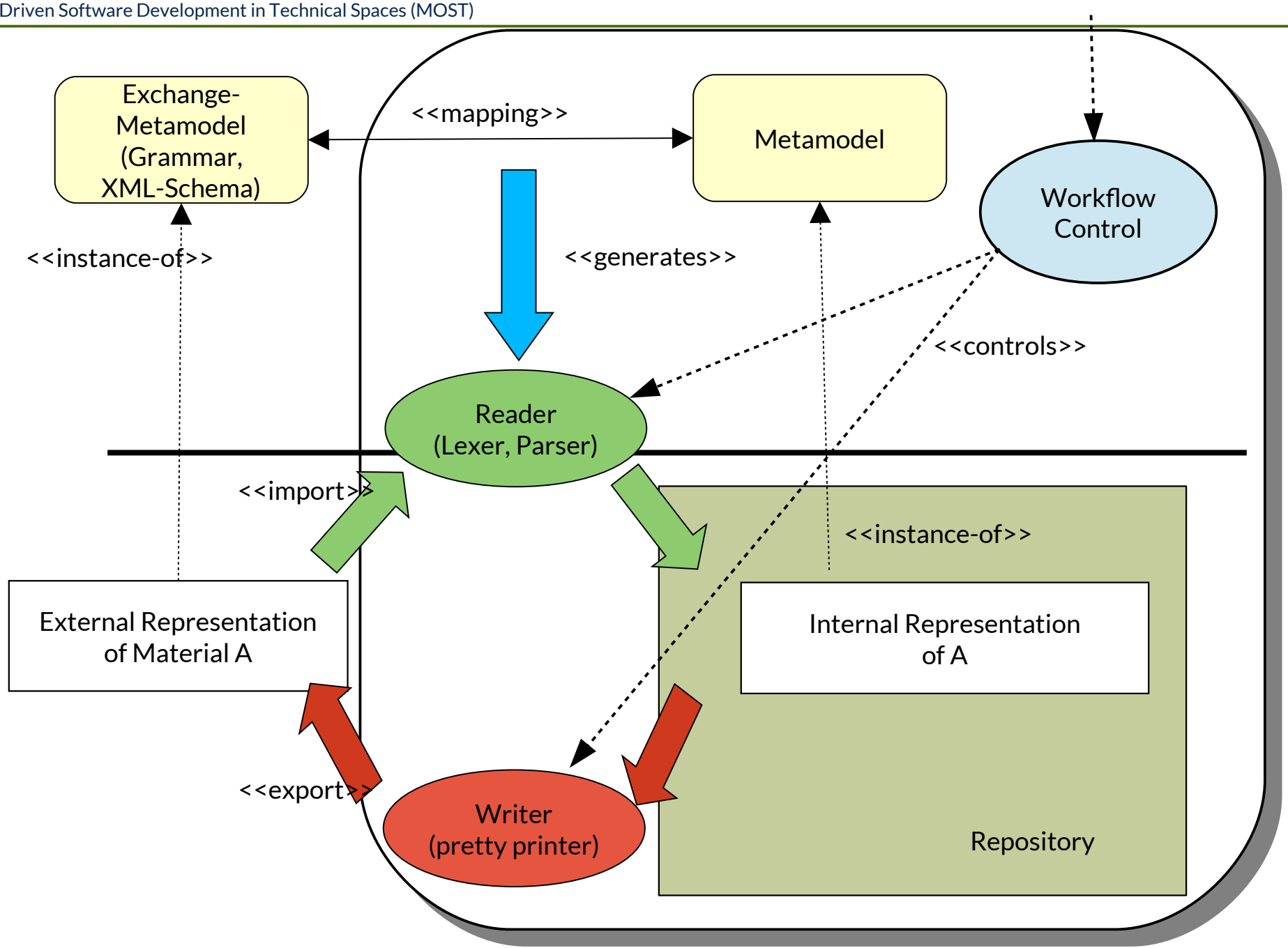
- From the metaclasses on M2, the structure of the types of the access interfaces on M1 (and many other code-packages) is derived



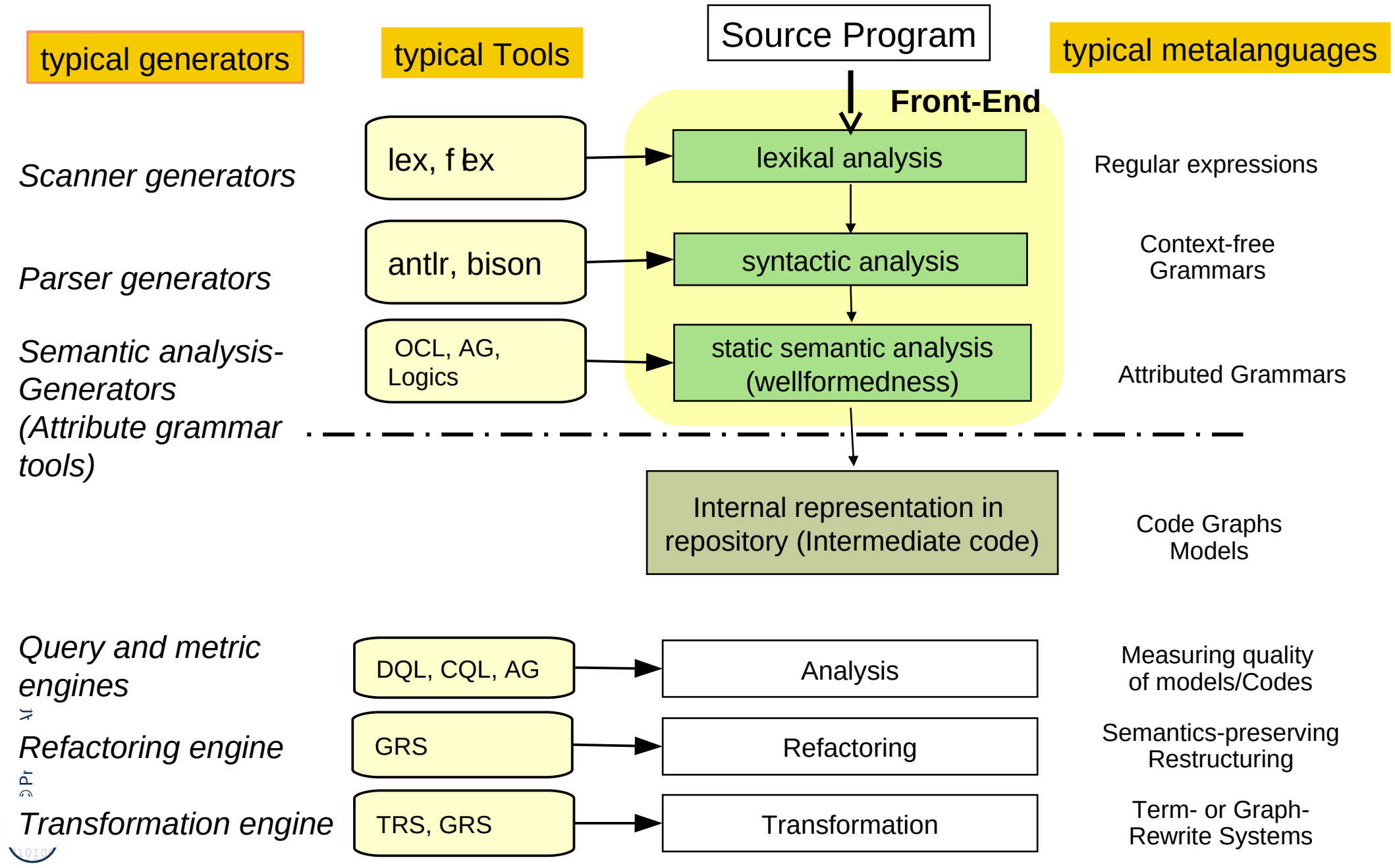
Generation of Interfaces and Packages in a Metamodel-Driven Repository

- Generation of **statically typed access interfaces** to the models and objects. **Typing and structuring** stems from metamodel:
  - Generation of **Access interfaces** for **Read und Write**, with implementation of Reader/Writer synchronisation- and transaction protocols
  - Generation of **Observer interfaces**: for observation of models and objects
  - Generation of **Reflective access interfaces** (weak typing or even untyped)
    - Access to object set (extend) of a clabject or the graphs of a model
- Generation of **Query interfaces** to the models and objects
  - implementation of a query interpreter from a query language
- Generation of **Consistency checkers ("Administration" aspect)**, to check the context-sensitive structure of the models and objects (wellformedness, Wohlgeformtheit, Integrität)
- Generation of Navigation
  - **Visitor-Packages**: navigation over the models and objects (with Visitor Pattern) (from DDL)
  - **Navigation packages:** for more general navigation (depth-first, breadth-first, inside-out, outside-in, layer-wise, etc.)
  - **Reachability** interfaces
- Generation of **Data exchange interfaces ("Infrastructure" aspect):**
  - Generation of **Importers** und **Exporters**, to transform the Materials into exchange formats of external metamodels (with a technical space bridge)
  - Implementations of **persistent objects** (activation, passivation)

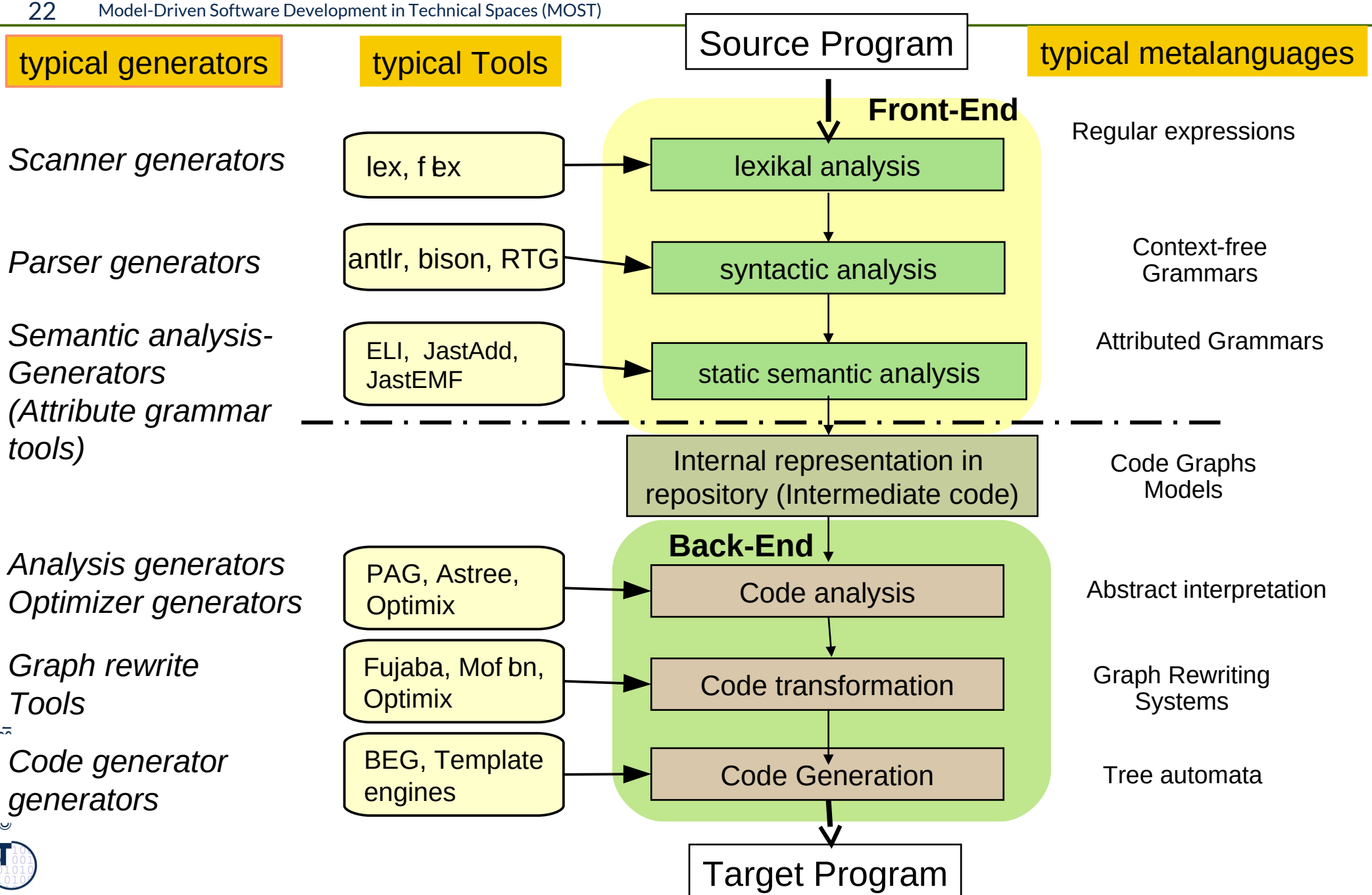
Use of Generated Importers and Exporters: How LinkTrees and Strings are used for Data Exchange



Q8: Phases of a Source Code Importers into a Repository and the Generating Tools; Example Refactoring Tool



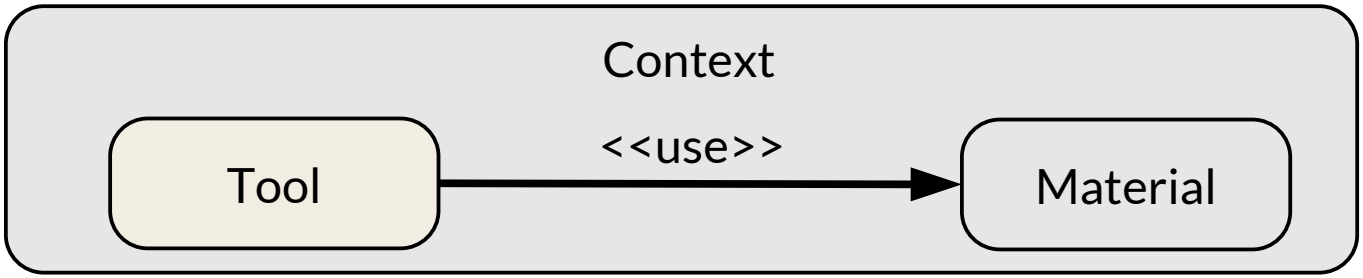
Q9: Phases of Compilers and Software Tools and Generators, Example Code Generator



42.3 Tools as Tool Objects accessing Materials in the Repository

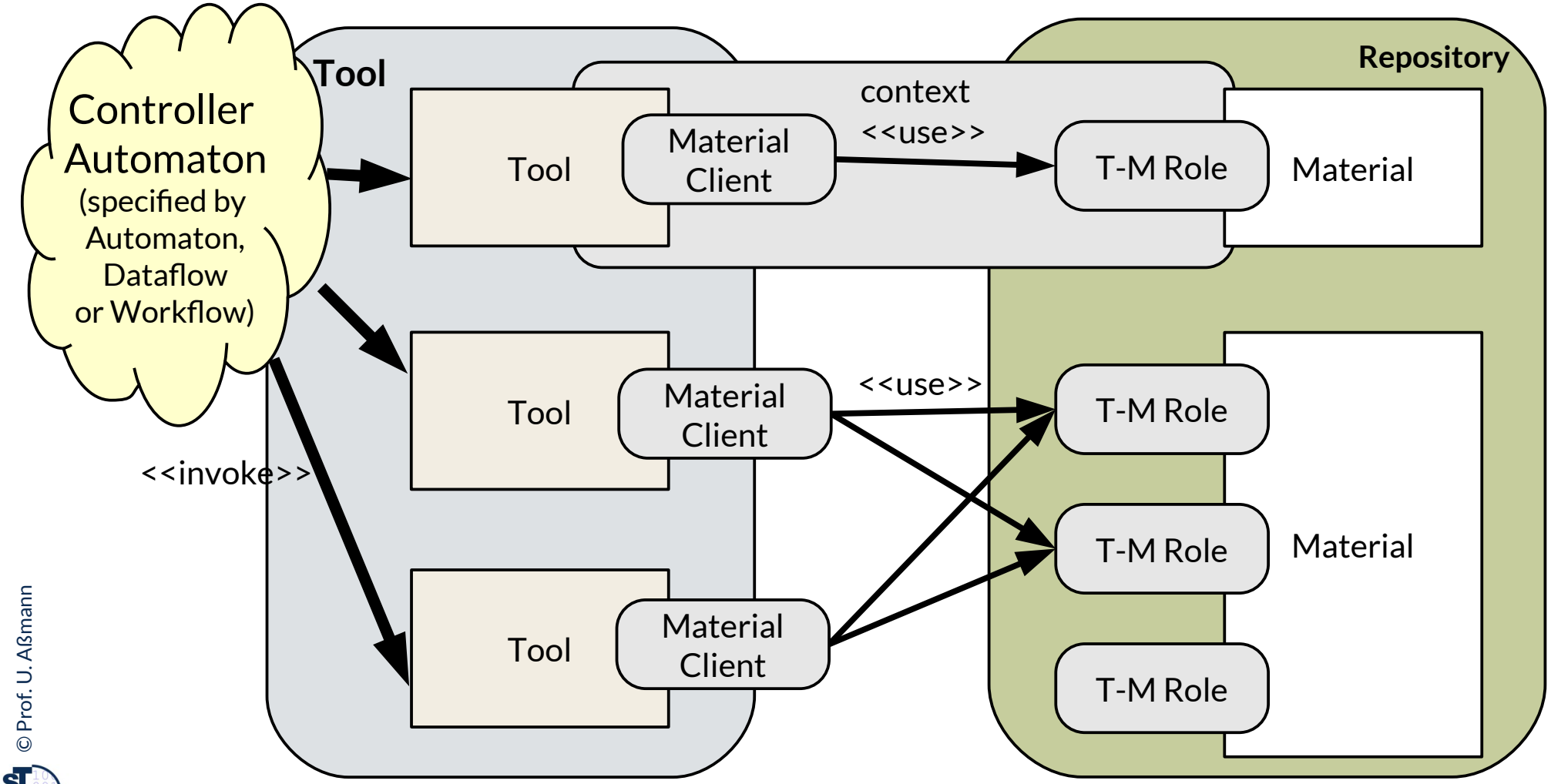
Tool-Material Collaboration Pattern

- A **tool-material collaboration** (T&M context-role model) expresses the *collaboration* of a tool and the material
  - The tool is active, has control; the material is passive and contains data
  - The tool has a view on the material, several tools have different views
  - It characterizes a tool in the context of the material, and the material in the context of a tool
  - The tool sees a role of the material, in collaboration with a tool
    - Roles of a material define the necessary operations on a material for one specific task
    - They reflect usability: how can a material be used?
    - They express a tool's individual needs on a material



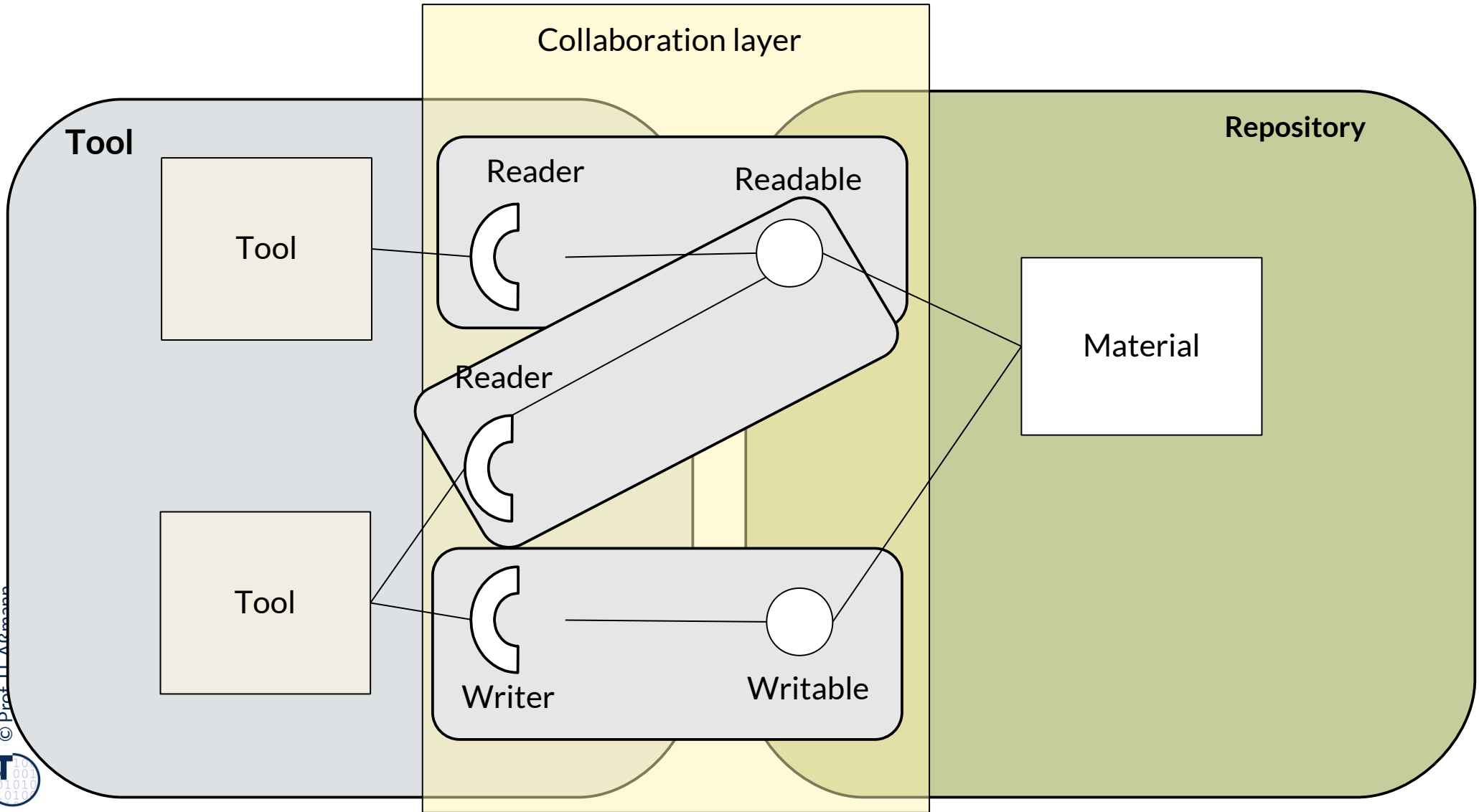
Controller Automaton, Tools and Their Views on Material

- Tools are controled by automata or workflows (TAM)
- Sometimes, we omit the context (if it is clear)



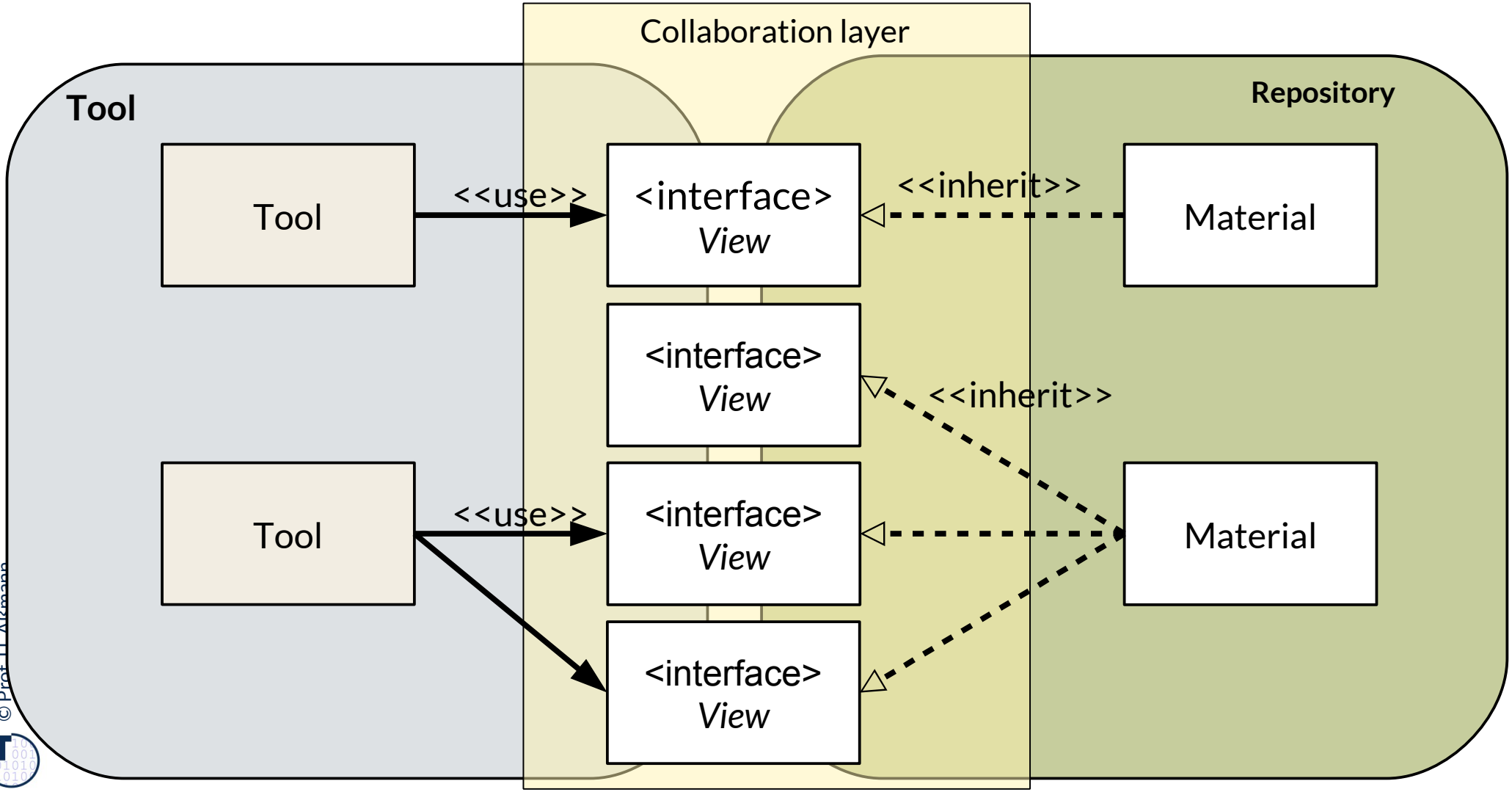
Collaboration Layer in UML Component Diagram Notation with Provided and Required Interfaces

- Roles can offer provided interfaces (represented by “lollipops”) or required interfaces (“plugs”)



How to Implement in Java? (Alt. 1)

- Roles can be realized by mixin inheritance or interface inheritance
- Then, roles of tools are integrated into tools

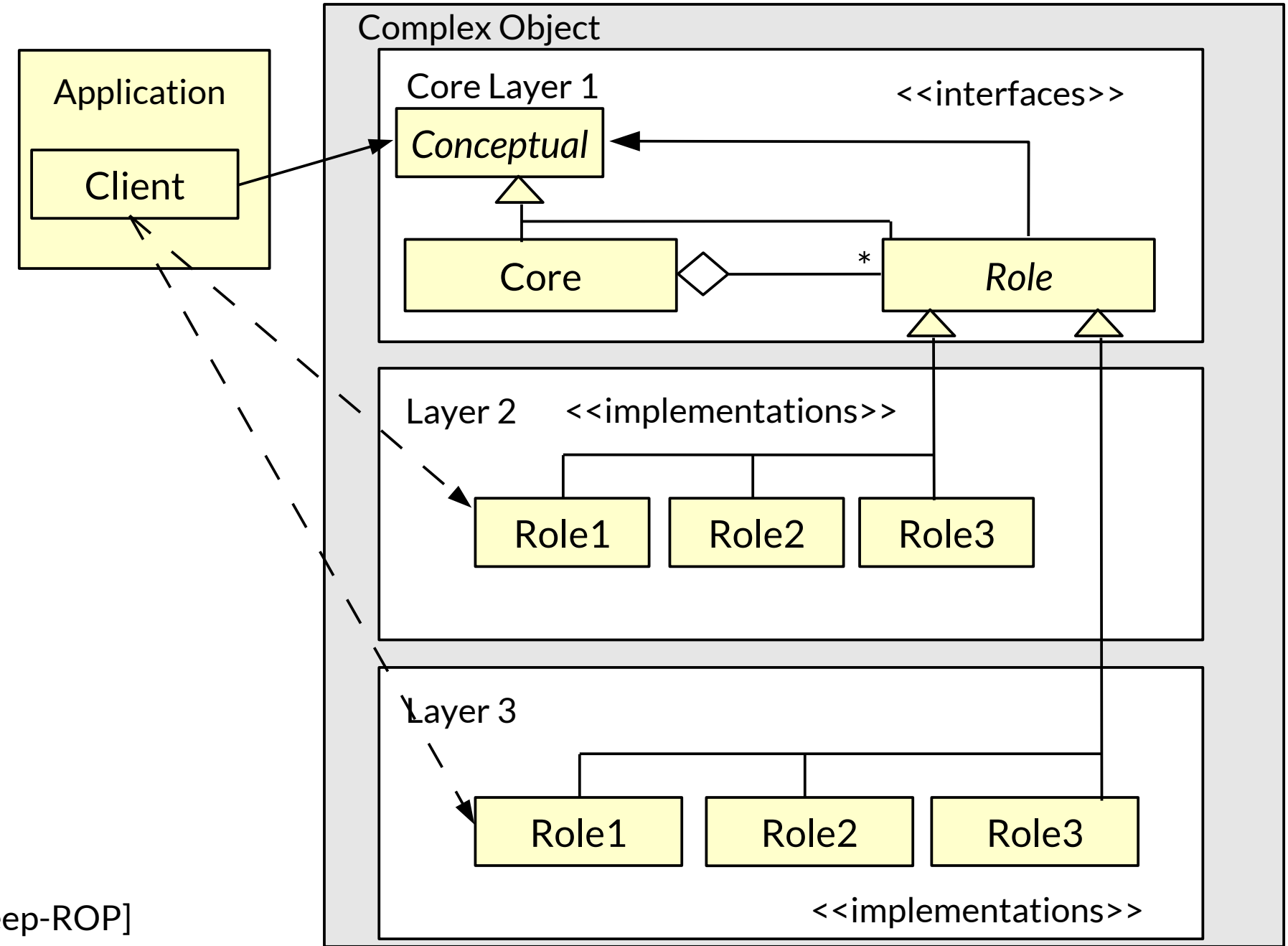


42.3.1 Layering of the Access Layers of the Material Objects with Deep Role-Object Pattern

Contexts and Layers

- A **layer** is a context of some roles which can be stacked on top of another layer
- The roles of a R.O.P-Object can be layered

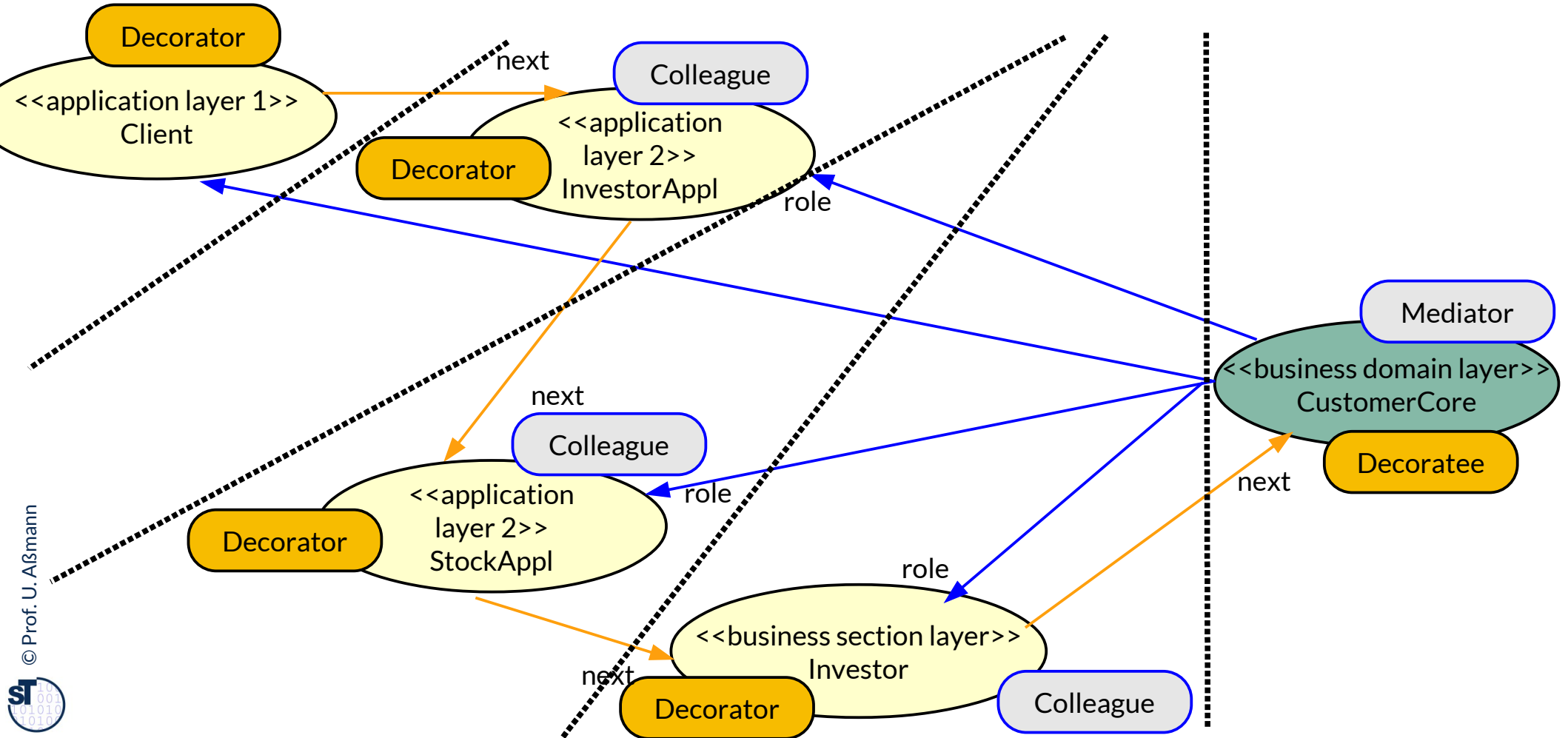
How to Implement in Java? (Alt. 2)
Riehle/Züllighoven's Deep Role-Object Pattern ("Deep Roles")



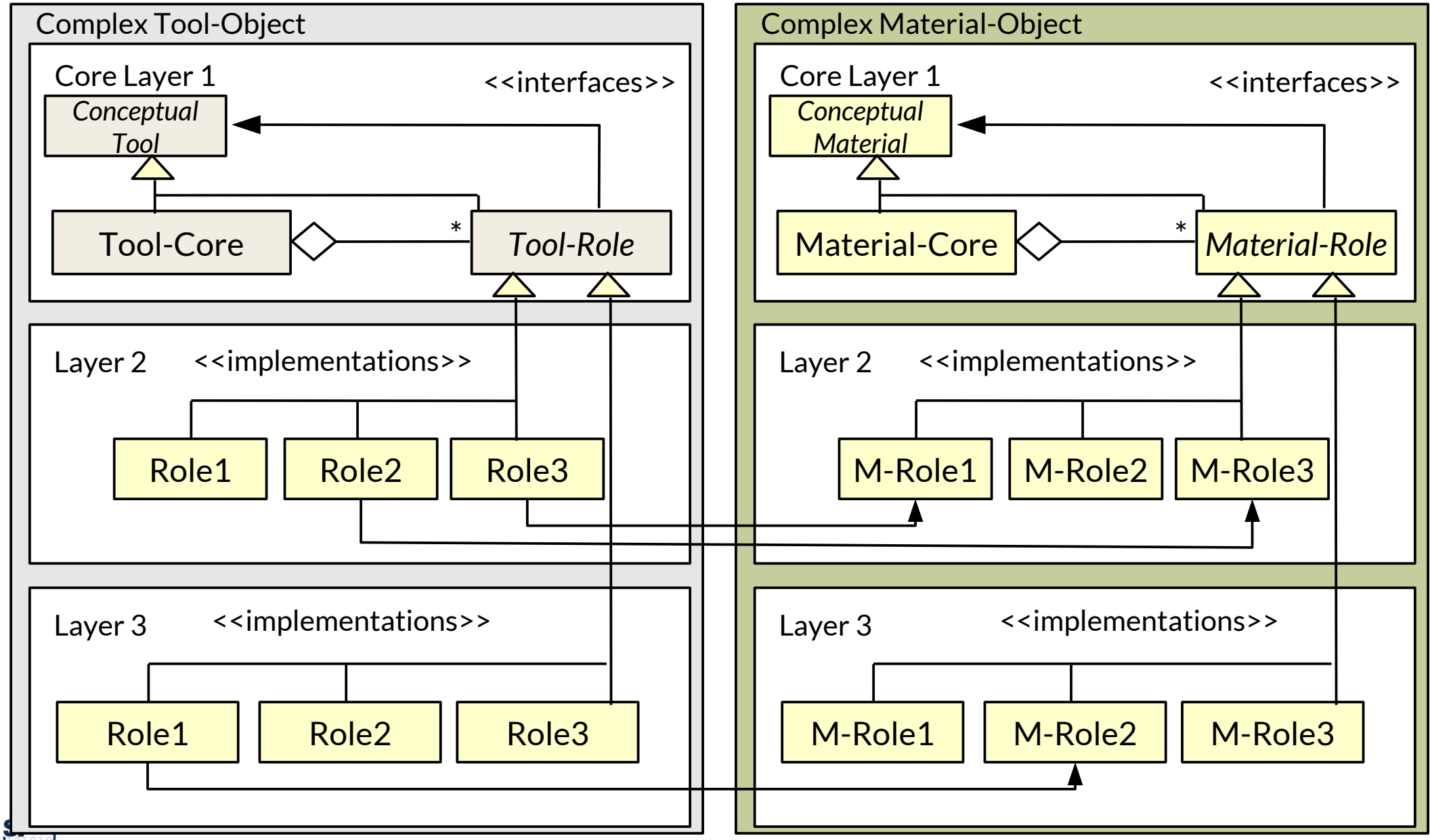
[deep-ROP]

Run-Time Structure of Deep R.O.P: Deep Roles as Decorators of a Core

- At runtime, RoleObjects pass service requests (queries) on to the core
  - RoleObjects can be stacked in a Decorator chain (**deep roles**)
- The core knows all RoleObjects, and distributes requests (Mediator)
  - The core manages the RoleObjects in a *map* that can be dynamically extended

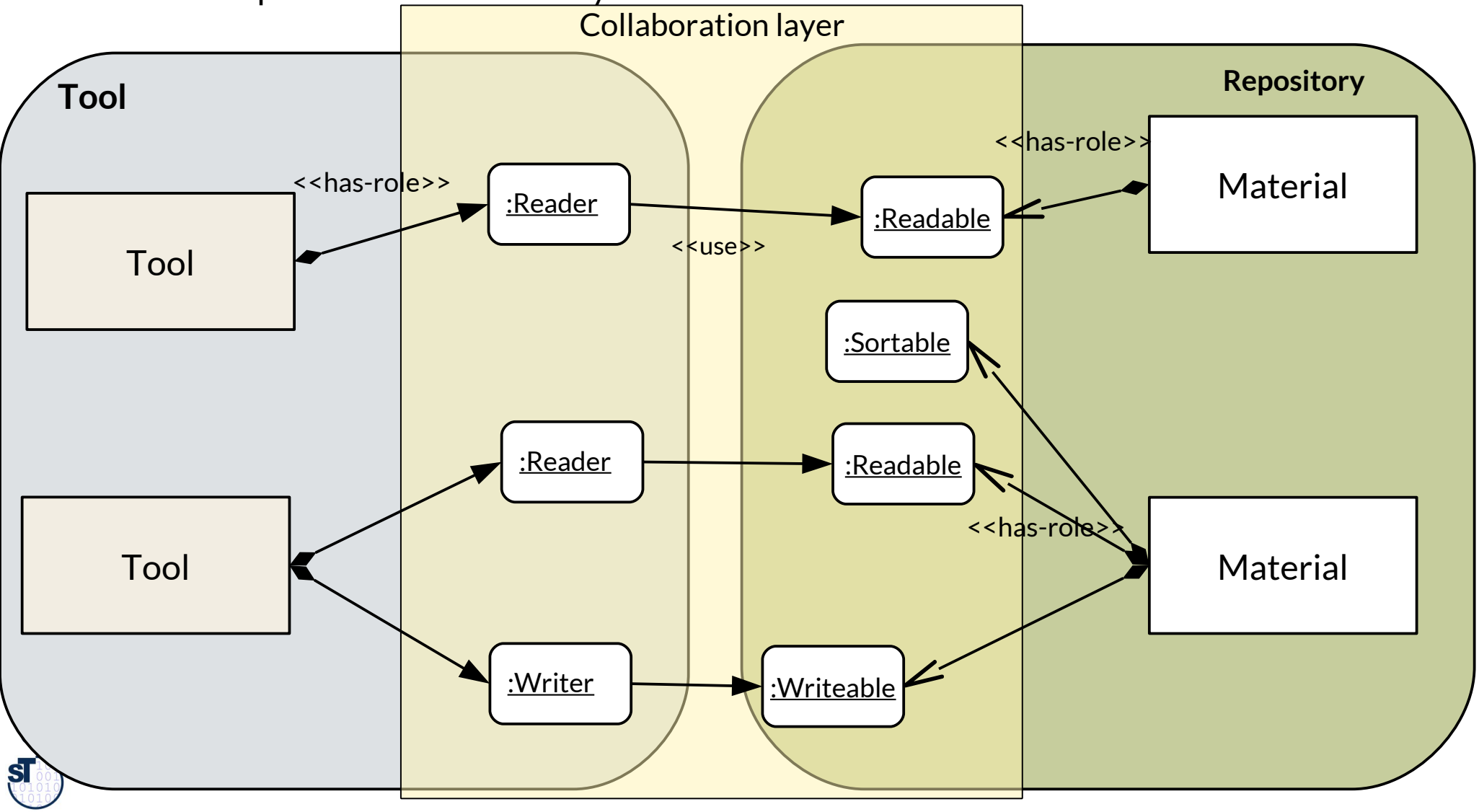


Deep Role-Object Pattern in Tool-Material-Collaboration



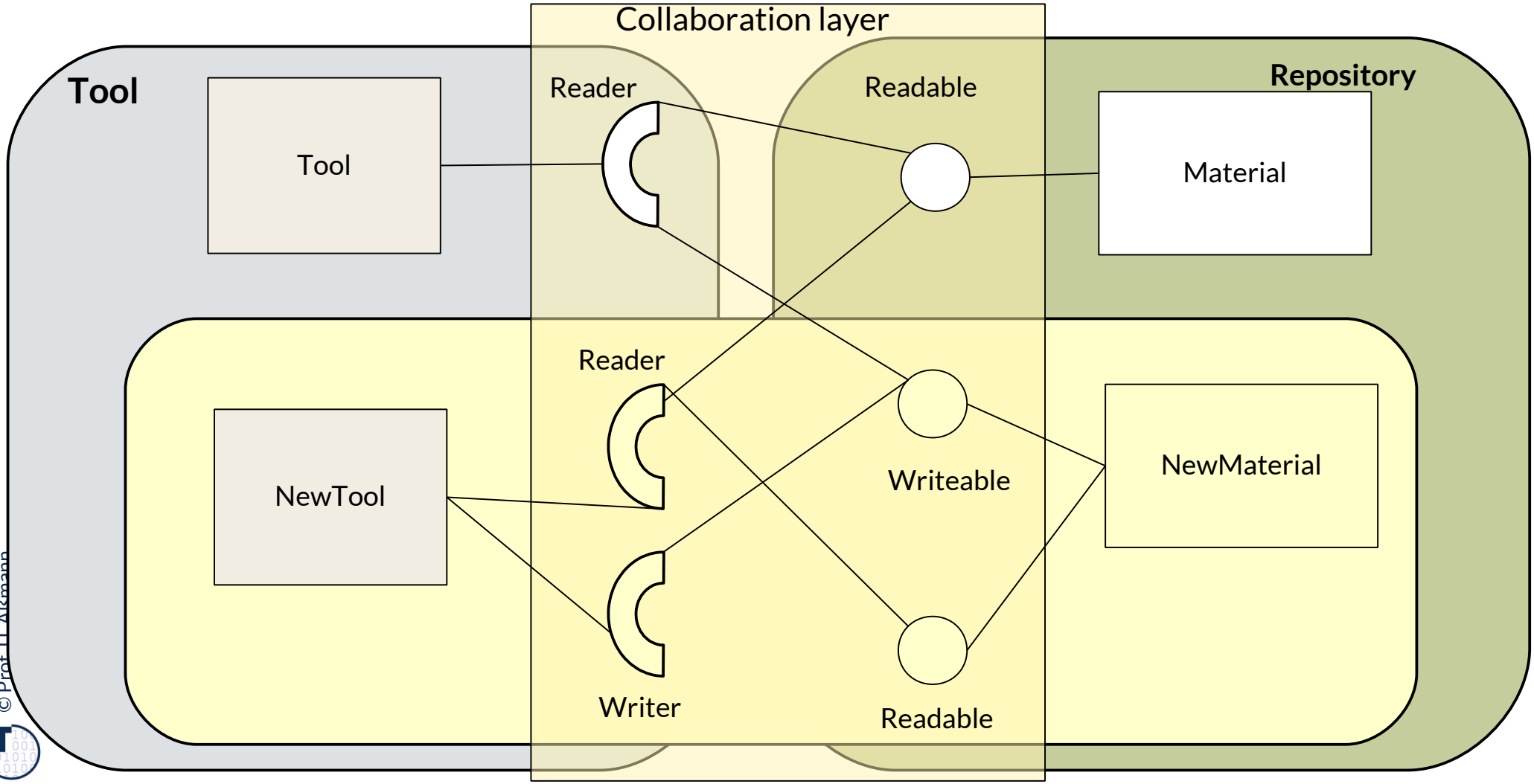
Implementation b) Tool-Material Roles as Role-Delegatees

- Roles can be realized by *role objects* (delegatees)
- Grouped in collaboration layers



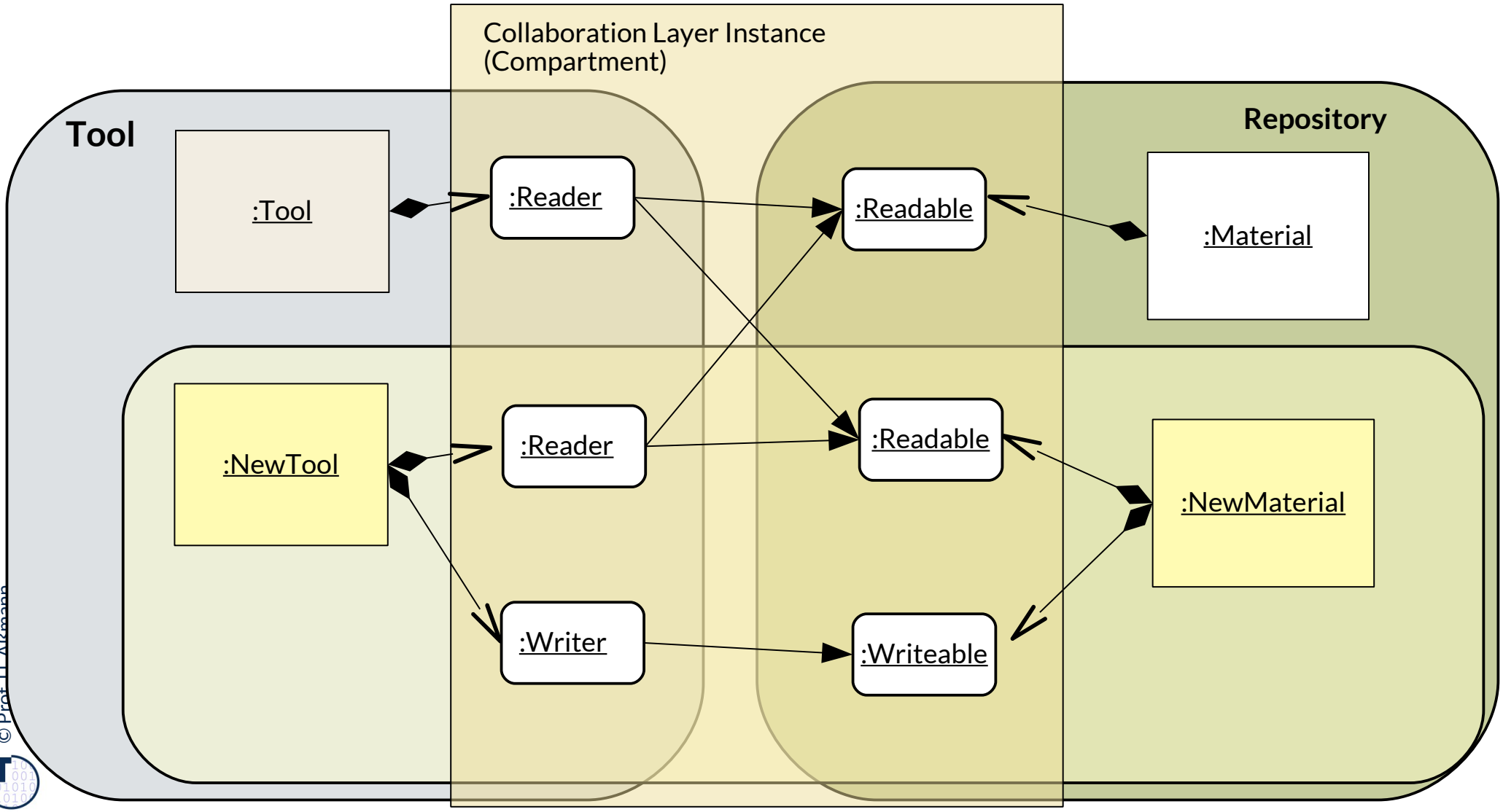
Extension of a Collaboration Layer

- Roles can be written down in UML as required and provided interfaces
- If a set of tools and materials is extended, new relations in the collaboration layer can be created. Here: Extension of the class net



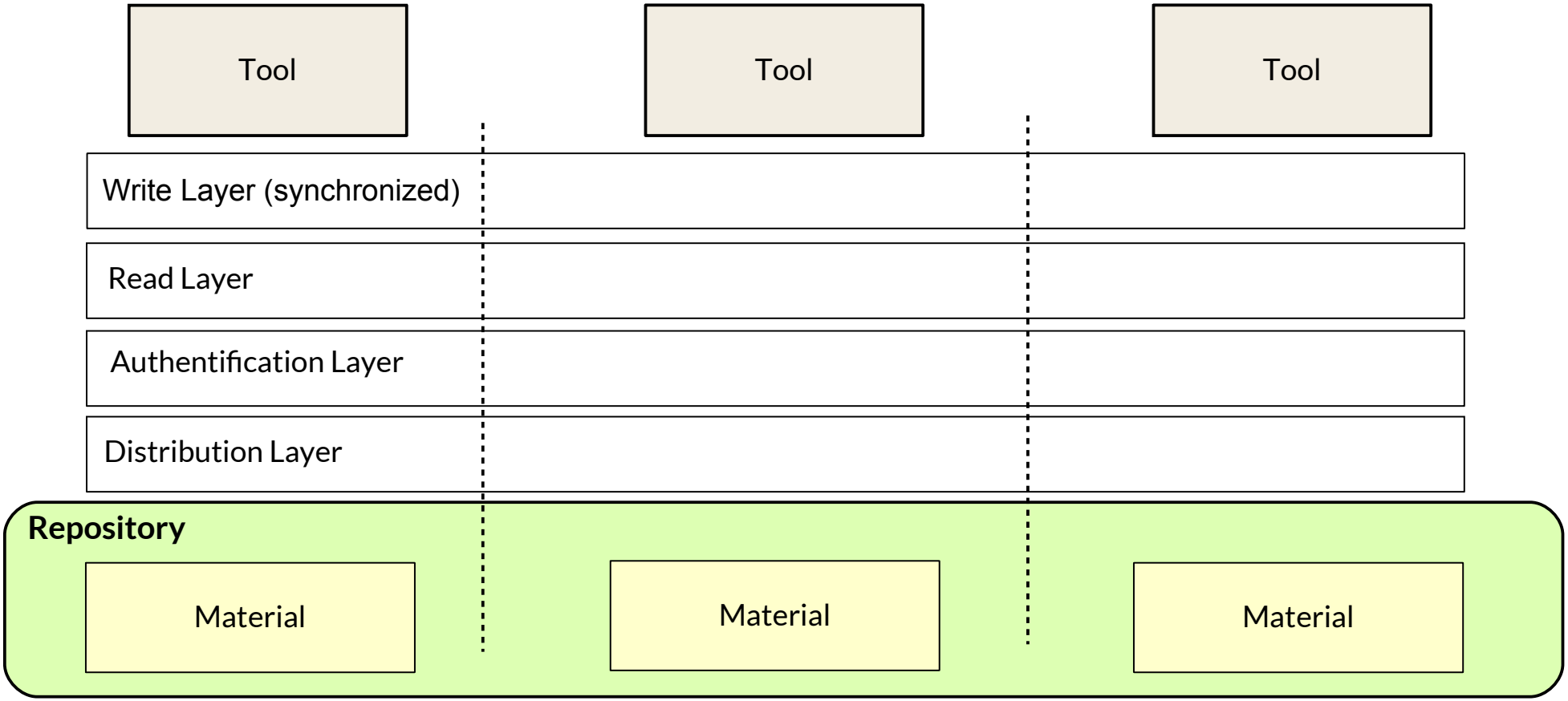
Extension of the Object Net in the Collaboration Layer

- Here: Extension of the object net of the compartment



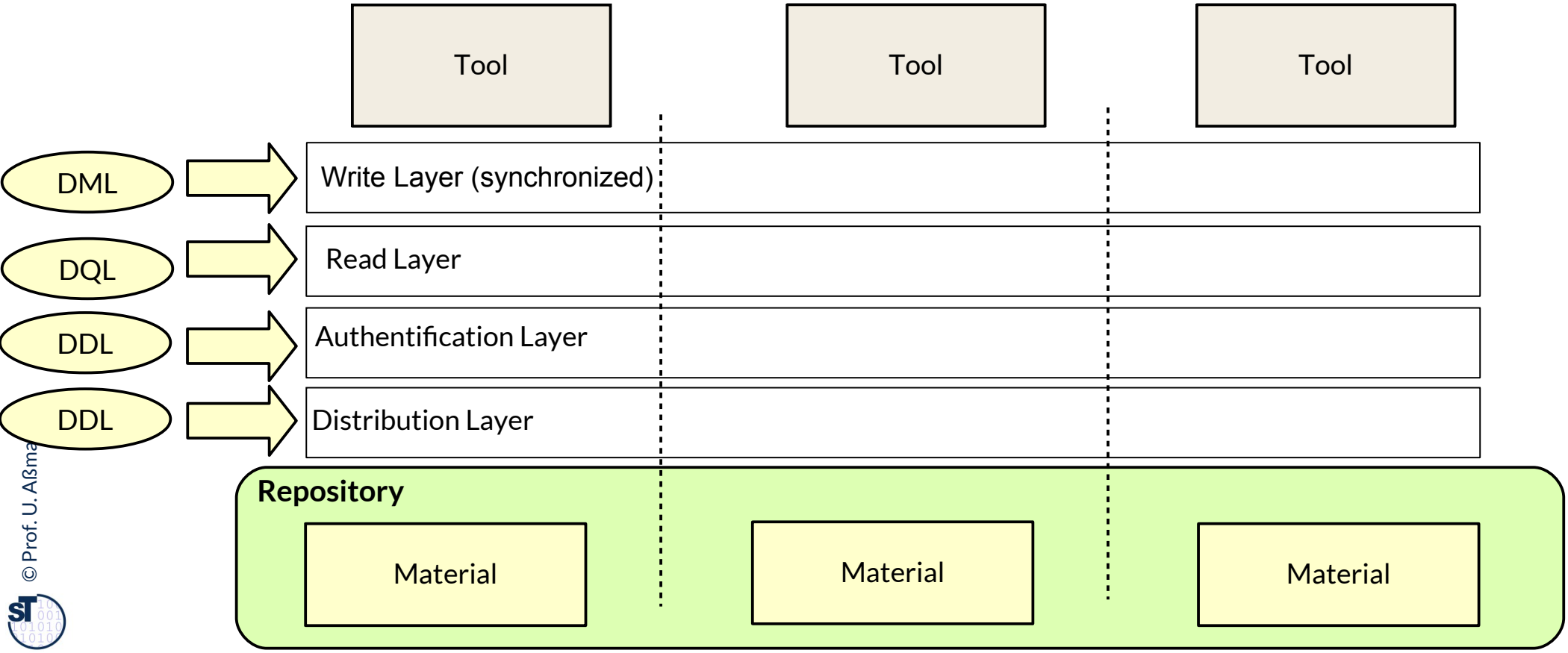
Access Layers of the Repository

- In a TAM-structured repository, Tool-objects access Material-objects through several layers of role objects
  - Every role object treats a specific aspect of the access
  - An Aspect-Material-object-Matrix results

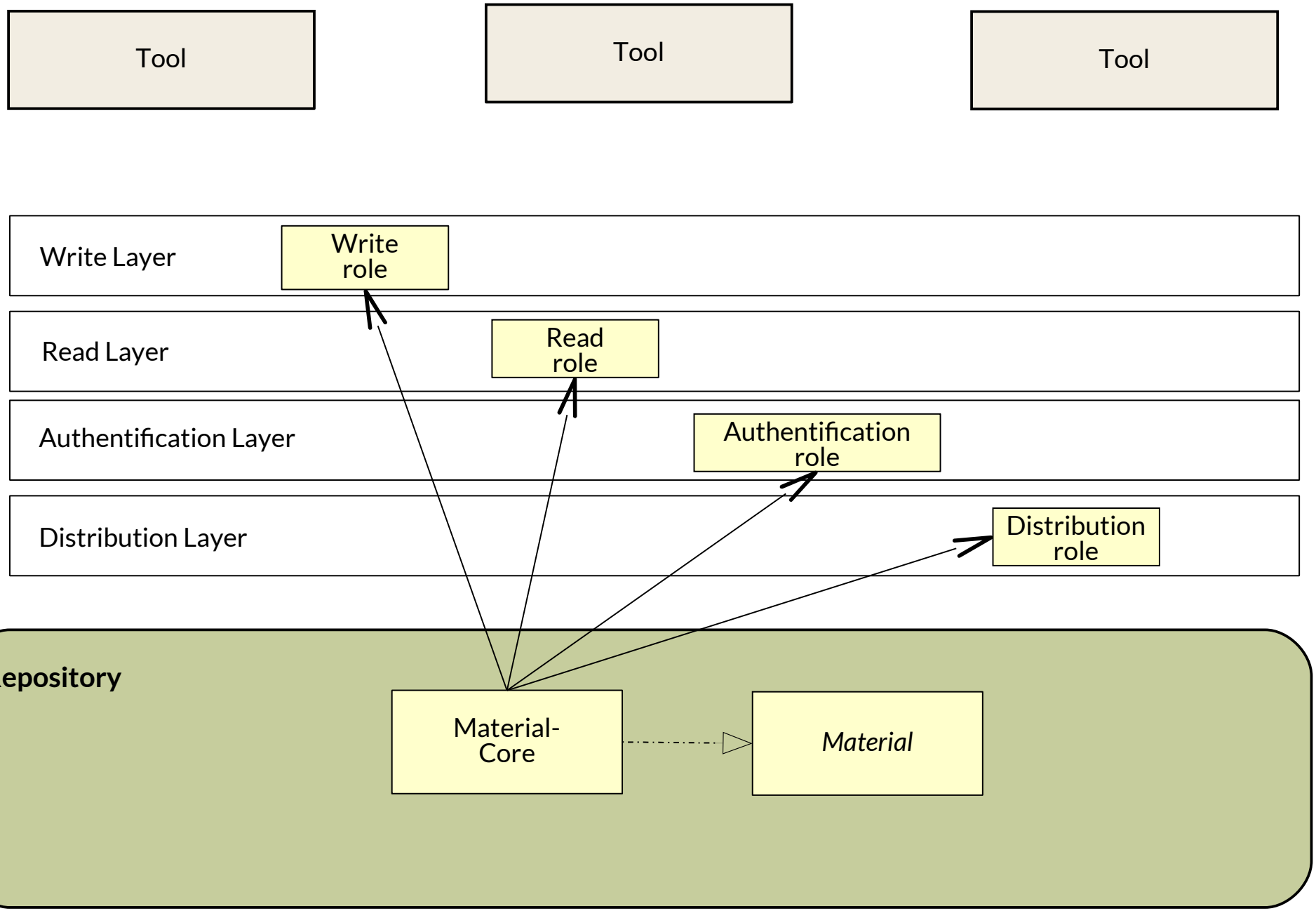


Generation of the Implementation of the Layers from M2

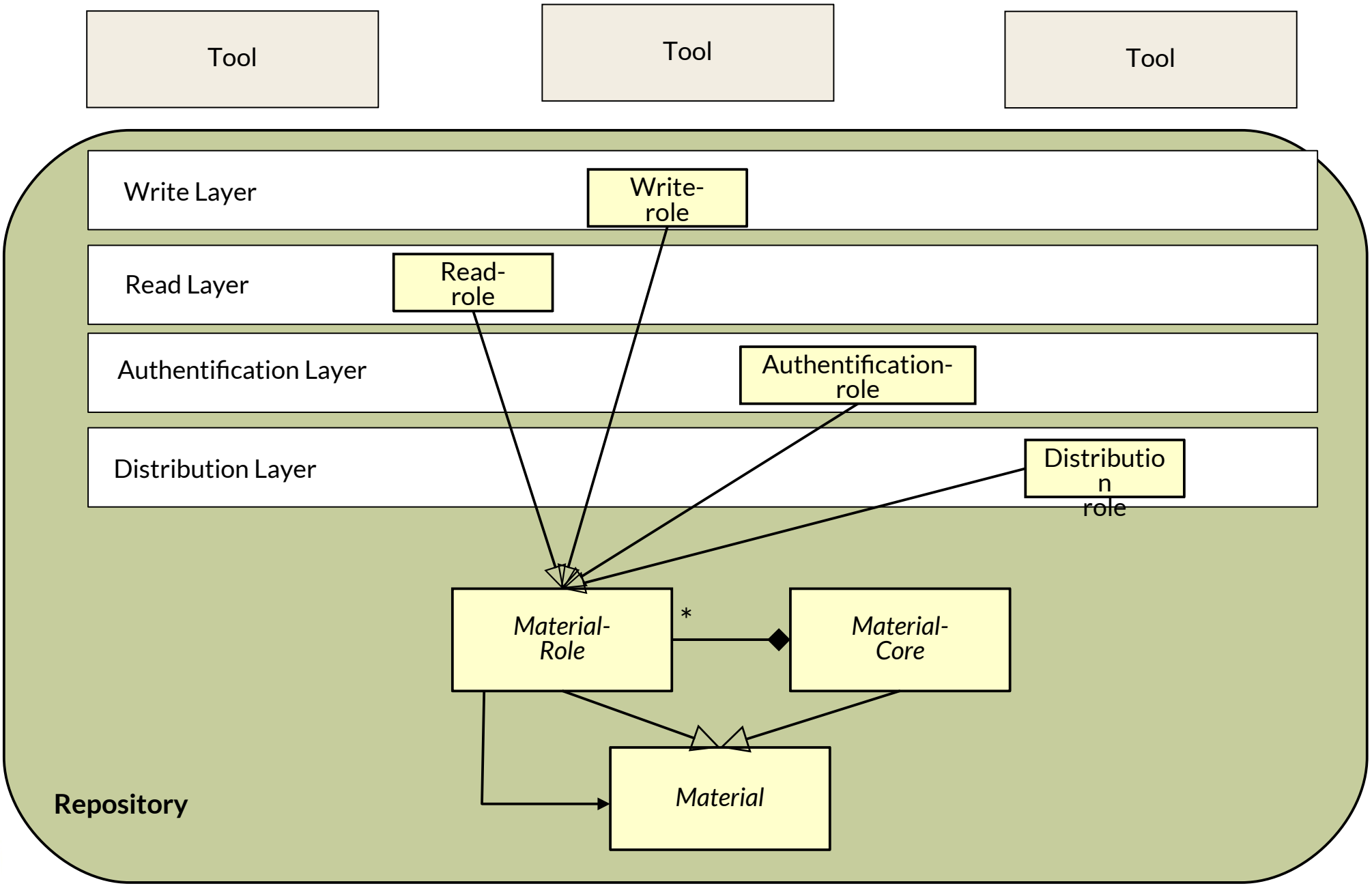
- The implementation of the role objects in the layers can be specified by different languages on M2
  - OCL-constraints can specify constraints in the Read/Write Layers
  - Query languages can be used to generate the Read Layer
  - Transformation languages can be used to generate the Write Layer



Implementation b) Material, realised as Core with Role-Delegatees (object-schizophrene)

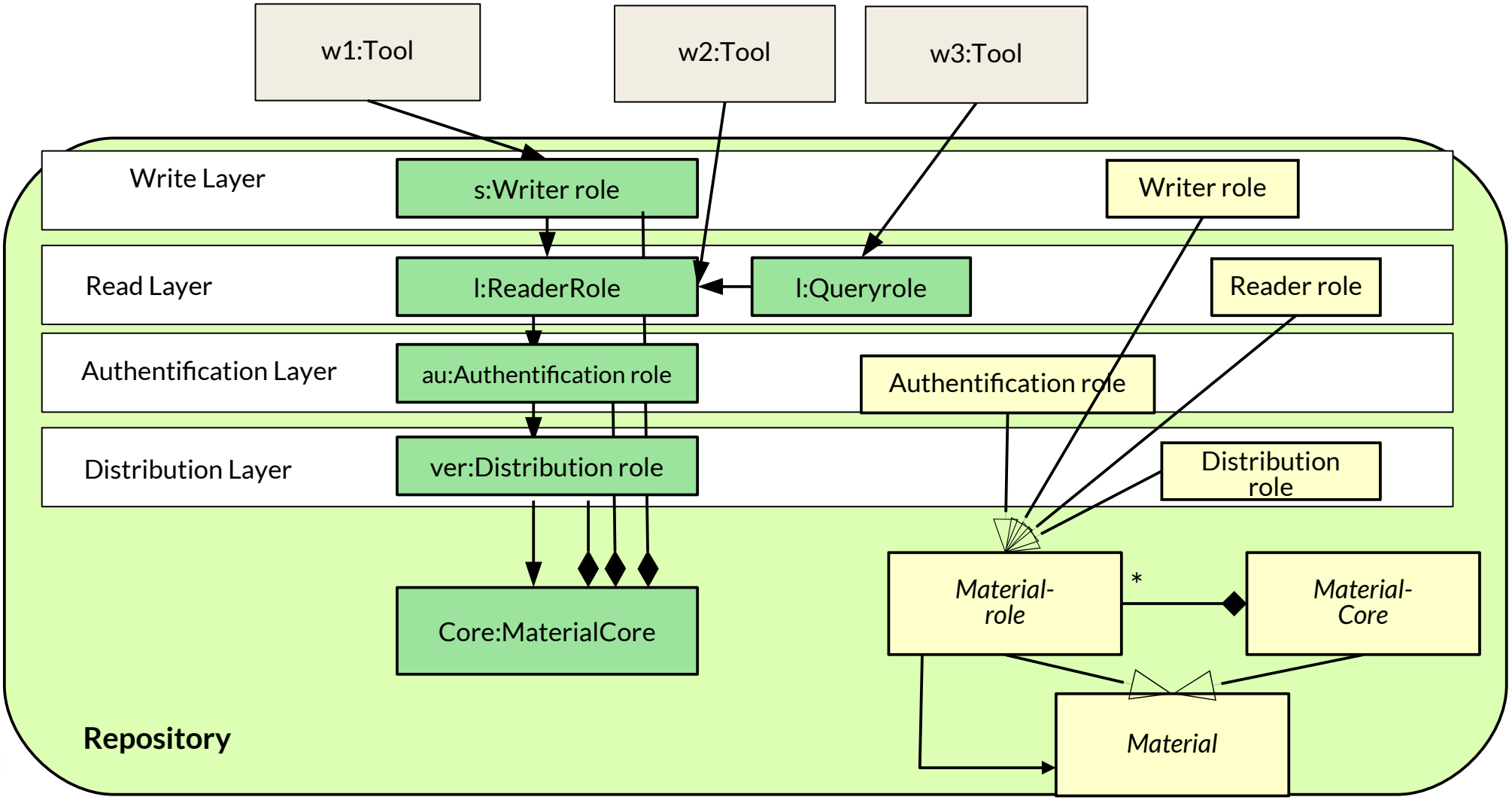


Implementation c) Material Structured with Deep Role-Object-Pattern (non-object-schizophrene)



Collaboration of Tool-Objects on a TAM-Repository with Role Objects

- The Role-Object Pattern is a good pattern for implementation of Tools and Materials in Repositories
- Collaboration layers (here only Material Layers are shown)
- Interfaces and implementations of these layers can be generated



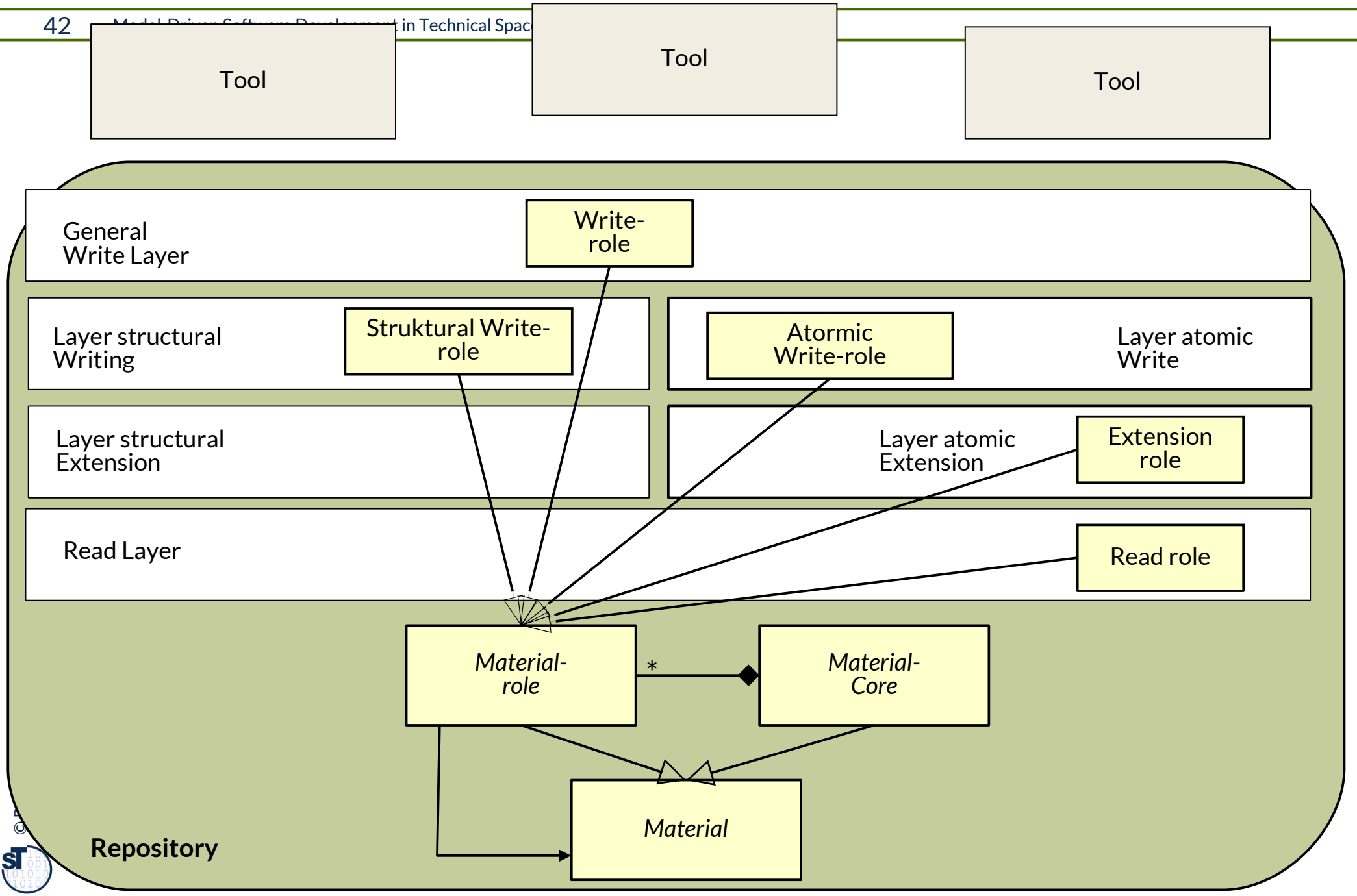
Tool-Effekte auf Materialien und Ihre Prägung für roles

- CRUD-Interface of Materialien in Informationssystemen bietet die Funktionalitäten:
  - **C**reate, **R**ead, **U**pdate (Modification), **D**elete

CRUD wird im Folgenden leicht erweitert:

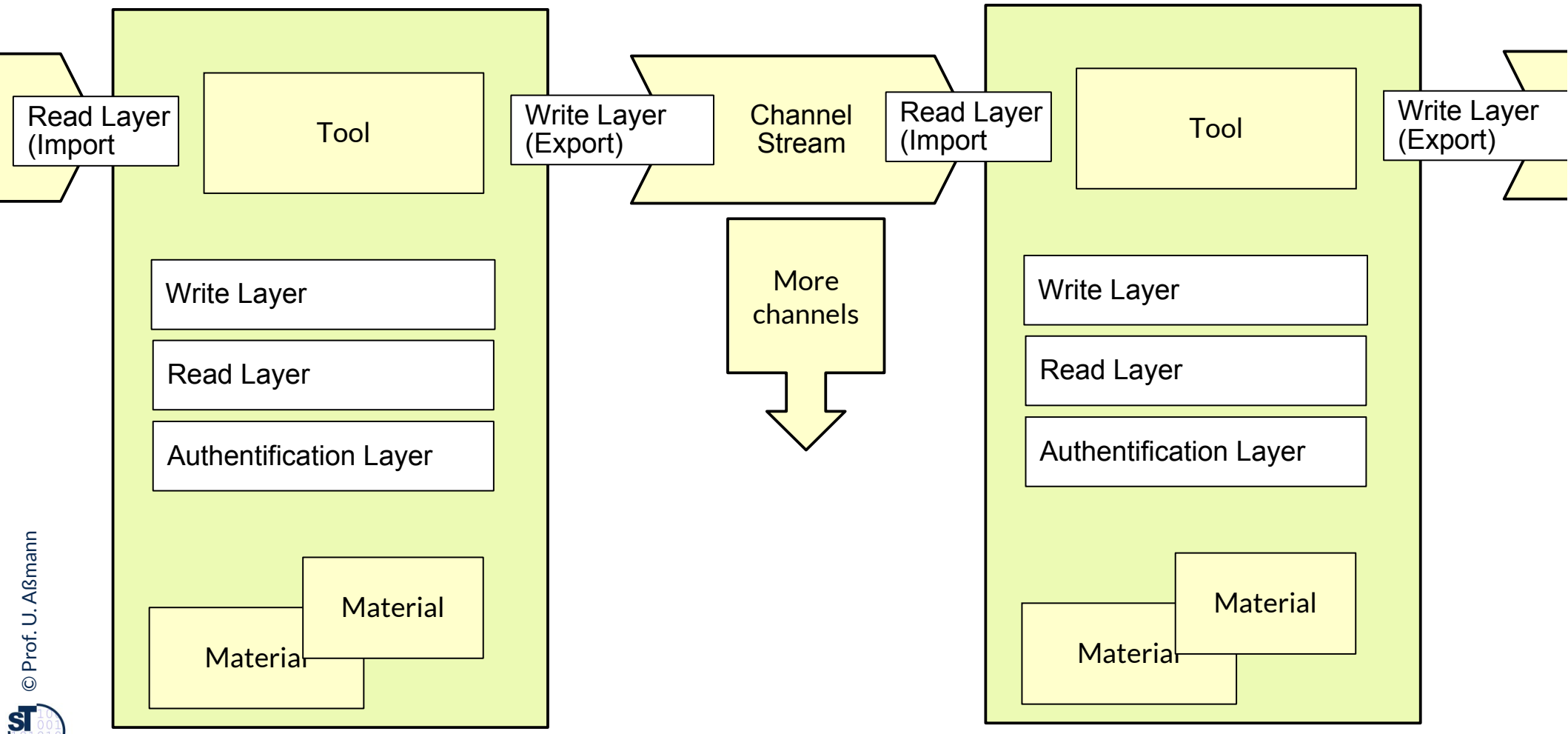
- Erweiterungs-role (**E**xtend)
  - Erweitert den Zustand des Objektes, zerstört ihn aber nicht
  - Atomare Erweiterung
    - Erweitert den Zustand der atomaren Attribute des Objektes
  - Strukturelle Erweiterung
    - Erweitert die strukturellen Attribute des Objektes, d.h. Des Objektnetzes, ohne die atomaren Attribute zu modifizieren
- Modifikation (Update)
  - Atomare Modifikation
    - Ändert den Zustand der atomaren Attribute des Objektes
  - Strukturelle Modifikation
    - Ändert den Zustand des Objektnetzes, ohne die atomaren Attribute zu modifizieren
- Allgemeine Modifikation
  - Alle obigen Effekte.

Other Possible Layers in the TAM Repository Architecture: More Layers Possible for other Effects on Materials

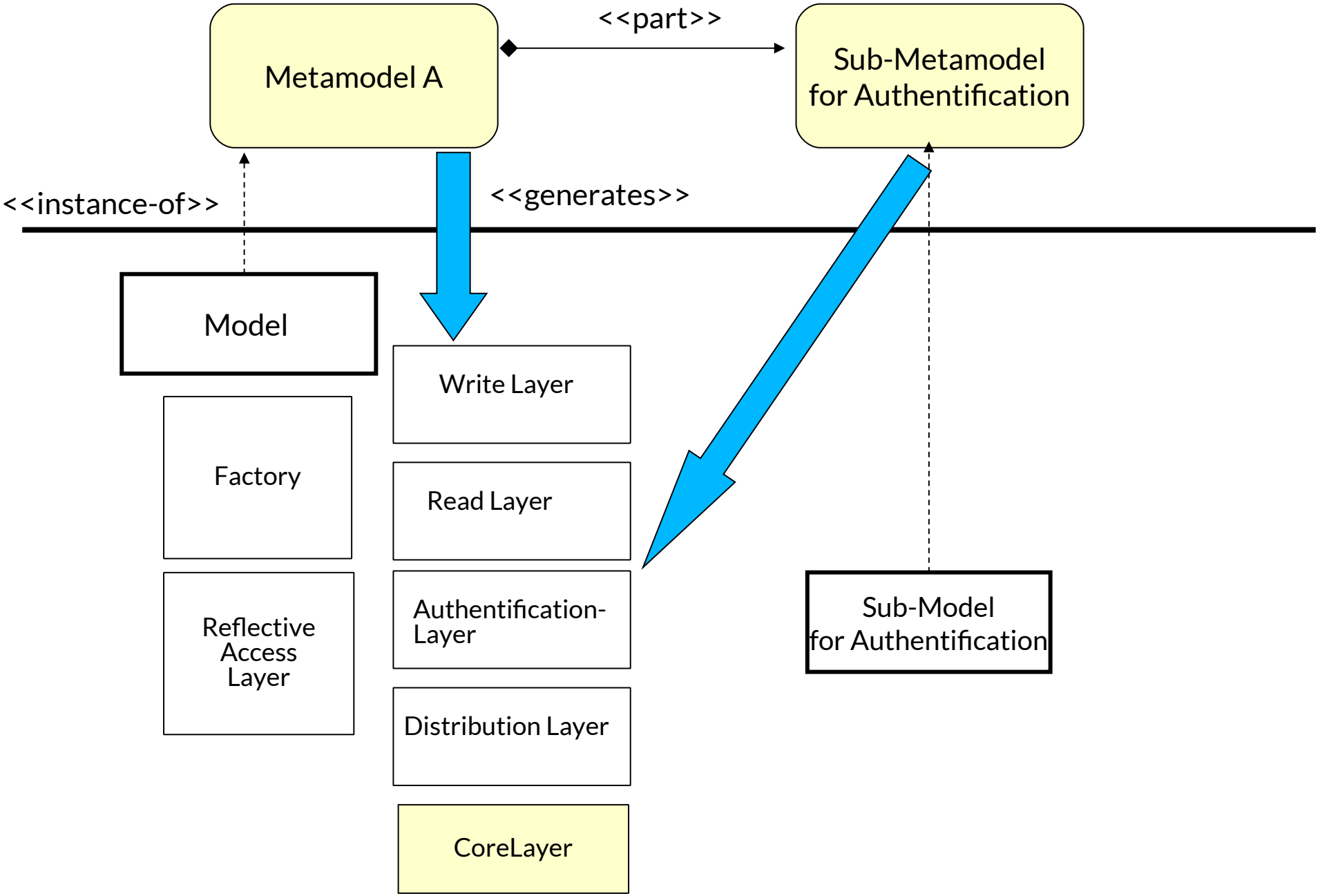


TAM for Stream-Based Tool-Architectures

- Is Tool embedded into a stream-based architecture, it has separate Read- und Write Layers for the input and output channels.
- One stream-based Tool alone is not distributed – the Distribution Layer vanishs.

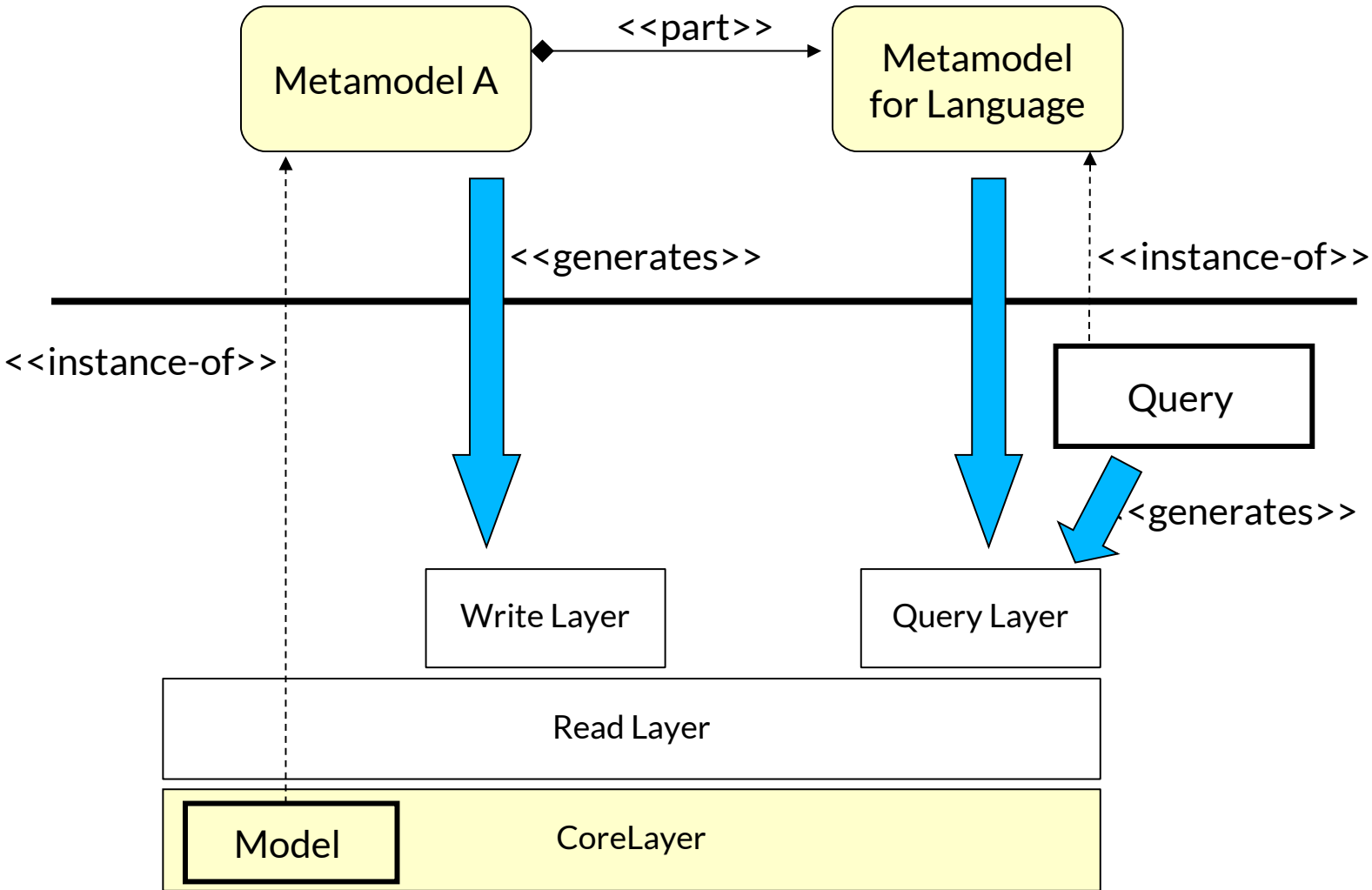


Generation of Access Layers from Metamodels



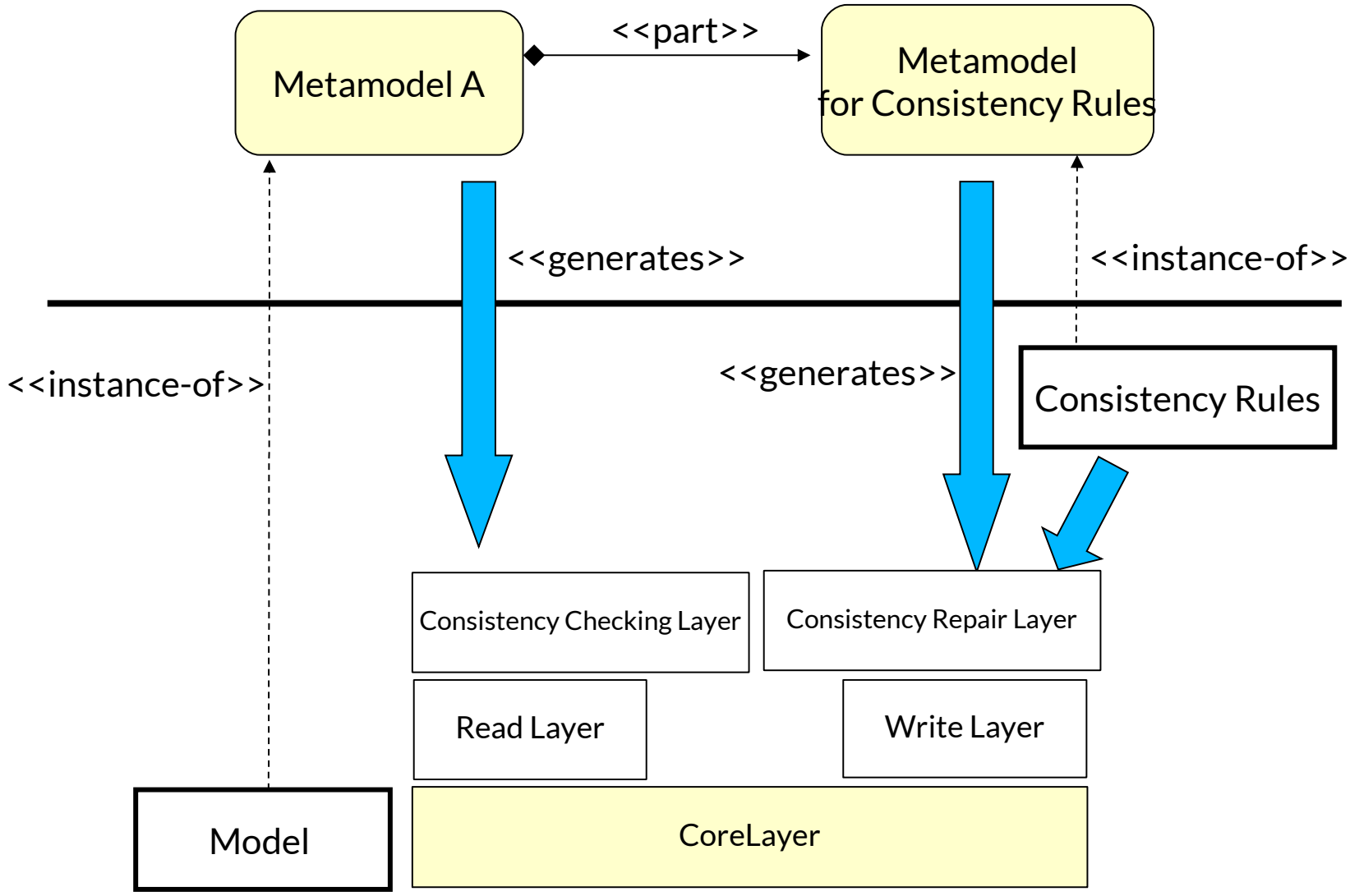
Generation of Query Layer from Metamodel of Query Language

- In a metamodel-driven repository, search, query, and reachability layers are generated from a query language (e.g., .QL)



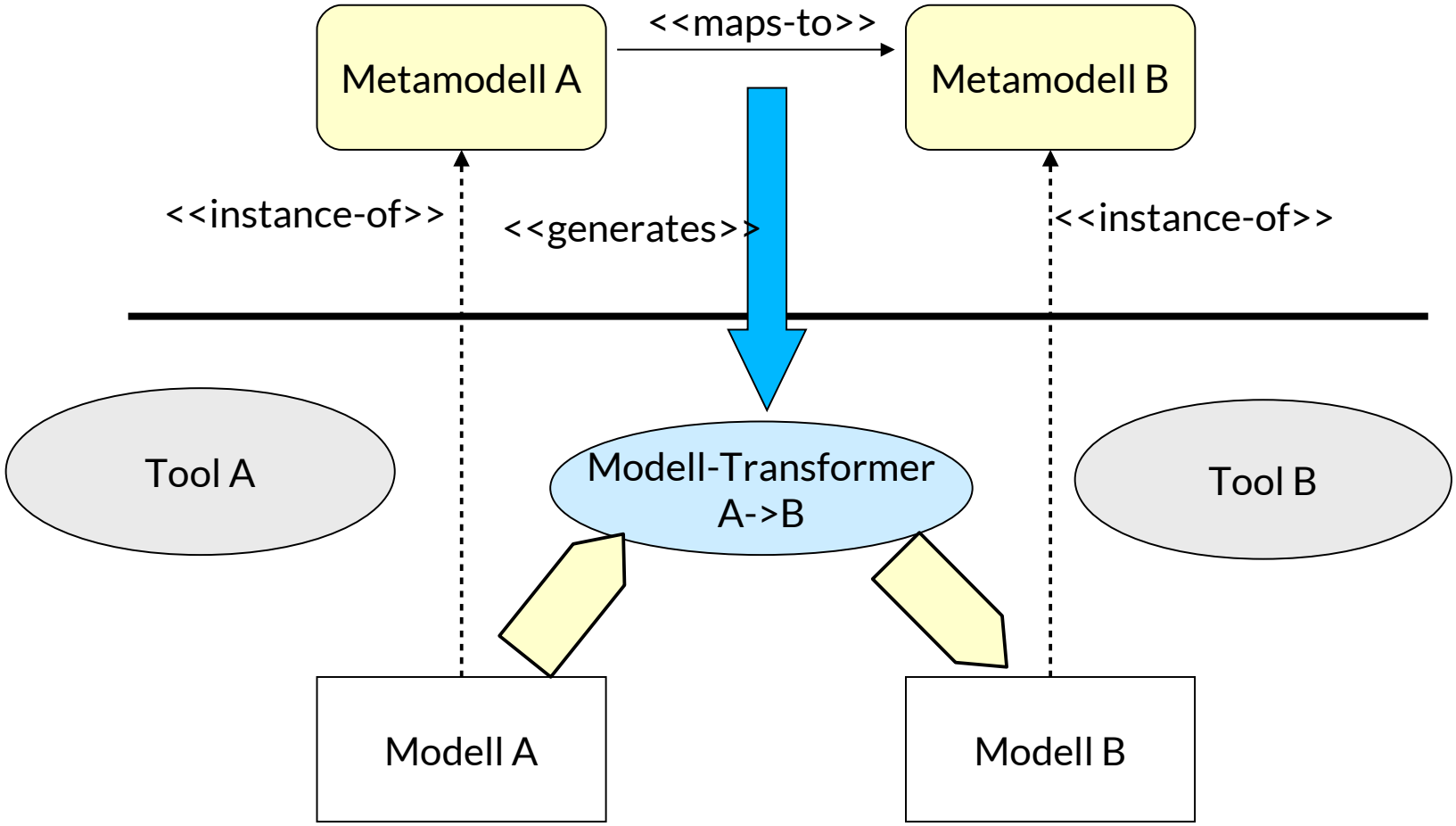
Generation of Administrative Layer

- In a metamodel-driven repository, consistency checking and repair layers ("administration aspect") are generated from a query language (e.g., .QL)



Generation of Model Transformer from Metamodel Mapping

- In a metamodel-driven repository,transformations (“administration aspect”) are generated from a query and transformation language (e.g., .Xcerpt)



42.4 Examples for Repositories on M0 and M1

Repositories can be used in different scenarios:

- Temporary for a Tool
- Persistent for a Tool
- Persistent for several Tools
- Persistent for entire company

Repositories can store object nets (M0), models (M1) and metamodels (M2)

File-Based Repositories

- No metamodels, but only simple metadata

Name	Characteristics
Microsoft Sharepoint	Web-filesystem-based repository
BSCW	Sharepoint made-in-Germany
WebDAV	Web protocol for distributed document management
Subversion	Version management systems on files and directories, usable from specific clients and browser One central repository
git, darcs	Version management systems on files and directories No central repository
DropBox, GoogleDocs	Cloud-based file system
Sparkleshare	Private-cloud-based file system, encrypted

Historical Examples for Repositories

Name	Characteristics
IBM Repository Manager	Repository für the IBM development process AD/Cycle, open architecture, team oriented
PCTE Object Management System	The repository of the first metamodel-based IDE in a techncial space, the PCTE environment
Pirol	View-based repository of TU Berlin (for token modeling) www.objectteams.org/publications/Diplom_Florian_Hacker.pdf

Company-Wide Repositories

Name	Characteristics
WebSphere Repository Database of IBM	Administration tools for Enterprise Applications
SAP R/3-System	Repository for Enterprise Applications, with R/3-Data-Dictionary for Metadata (relational schema)
SAP NetWeaver Master Data Management (MDM)	Distributed Repository for Enterprise Applications

Quelle: Habermann, H.-J., Leymann, F.: Repository - eine Einführung; Oldenbourg Verlag 1993

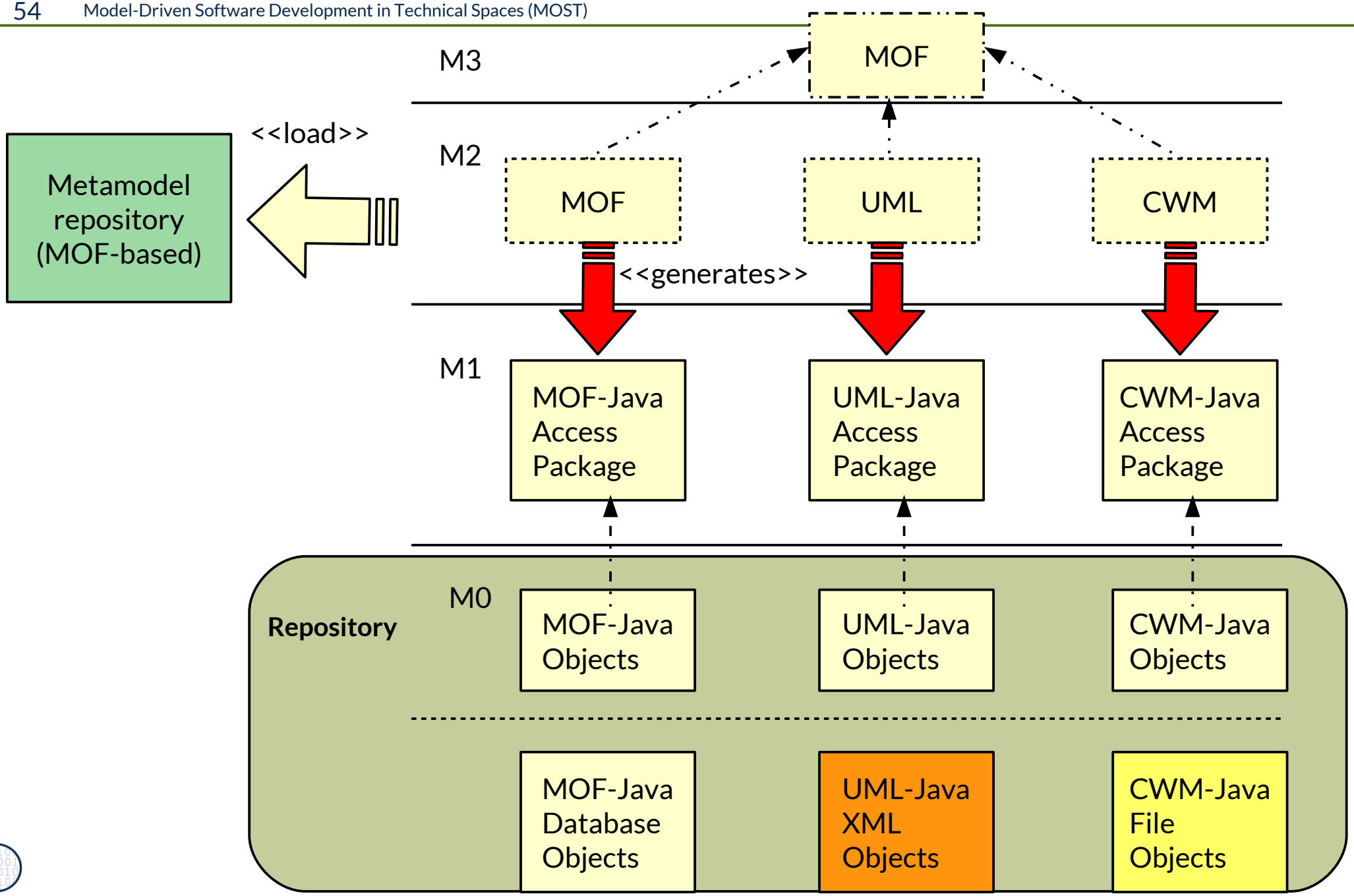
Metamodel-Driven Repositories

Name	Characteristics
Eclipse EMF	Metalanguage EMOF; similar to MDR
EMFText	based on EMF, with automatically generated transformers (importers, exporters to Grammarware)
Netbeans Metadata Repository (MDR)	Metalanguage MOF; Load of several metamodels possible möglich; Generation of typed access interfaces and reflective access interfaces; mapping to database and file system possible
ModelBus	Metalanguage MOF; Distributed repository für MOF-based models
Hibernate	Persistence module for object-oriented programs (Java) with object-relational mapper (ORM); mapping of Java-objects to relations and different SQL-databases

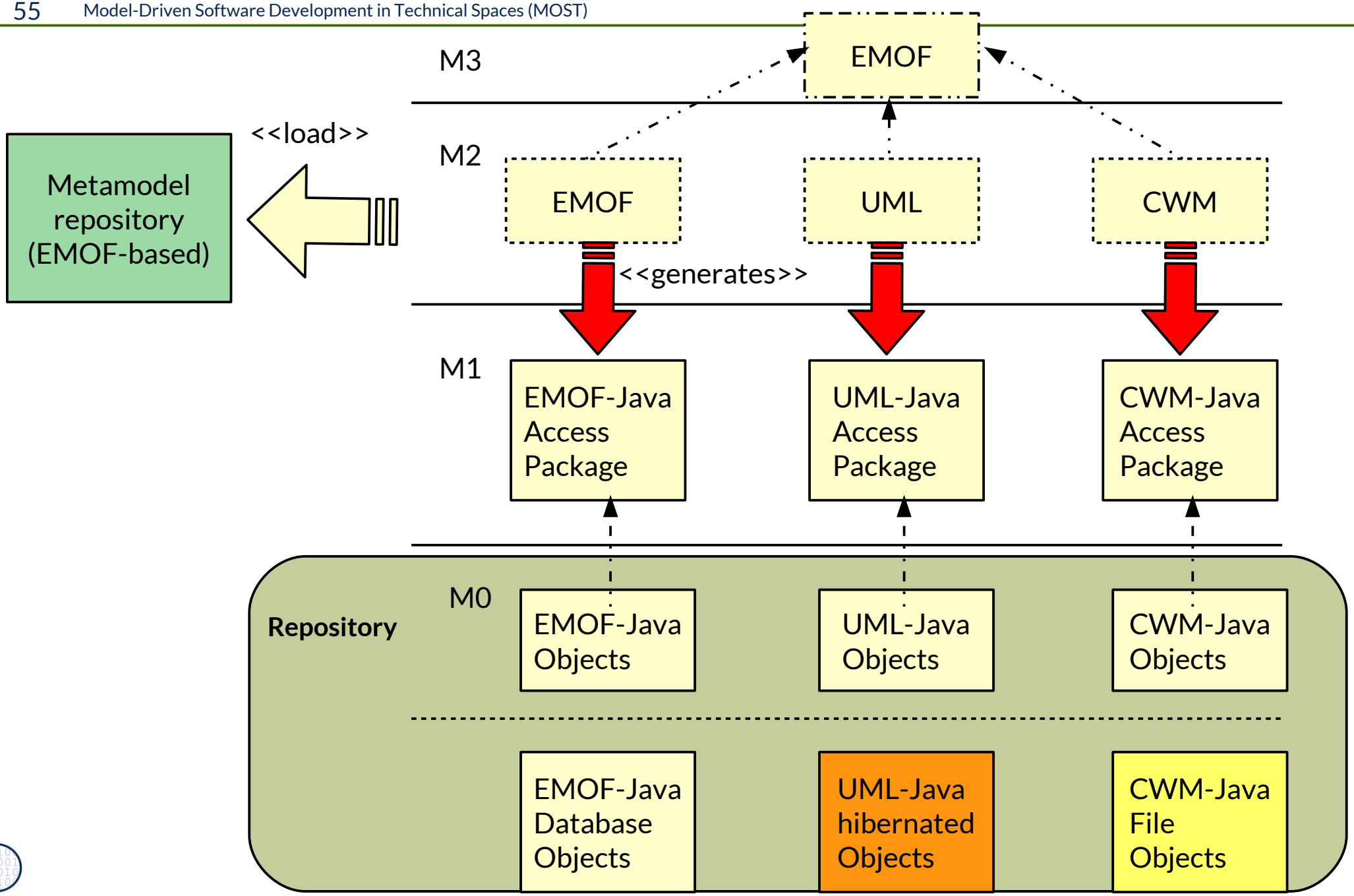
Netbeans MDR

- MDR is a metamodel-driven persistent repository for the IDE Netbeans (<5.5) with metalanguage MOF https://www.openhub.net/p/8317
- Martin Matula. (martin.matula@sun.com). NetBeans Metadata Repository.
  - https://netbeans-uml-extender-plugin.googlecode.com/files/MDR-whitepaper.pdf
  - https://netbeans.org/download/5_5/javadoc/org-netbeans-api-mdr/overview-summary.html
- Storing MOF-metamodels (metadata repository), also models (model repository), and object-nets (graphs)
- Advantages:
  - May be used distributedly, company-wide
  - Generation of Java-access interfaces via JMI (MOF-Java mapping)
  - Generation of reflective access interfaces (weak typing)
  - Generation of observation interfaces
  - Transparent storage in main memory, filesystem or database
  - Exchange format XMI (MOF-XML mapping)

Netbeans MDR



Eclipse EMF (based on EMOF)



Model Management in the Distributed Repository ModelBus

- Www.modelbus.org
- Started in France at IRISA Rennes, now hosted at FhG Focus Berlin

Quelle: Blanc, X., Gervais, M.-P., Sriplakich, P.: Model Bus: Towards the Interoperability of Modelling Tools; in Aßmann, U. u.a.: Model Driven Architecture, Springer Verlag 2005

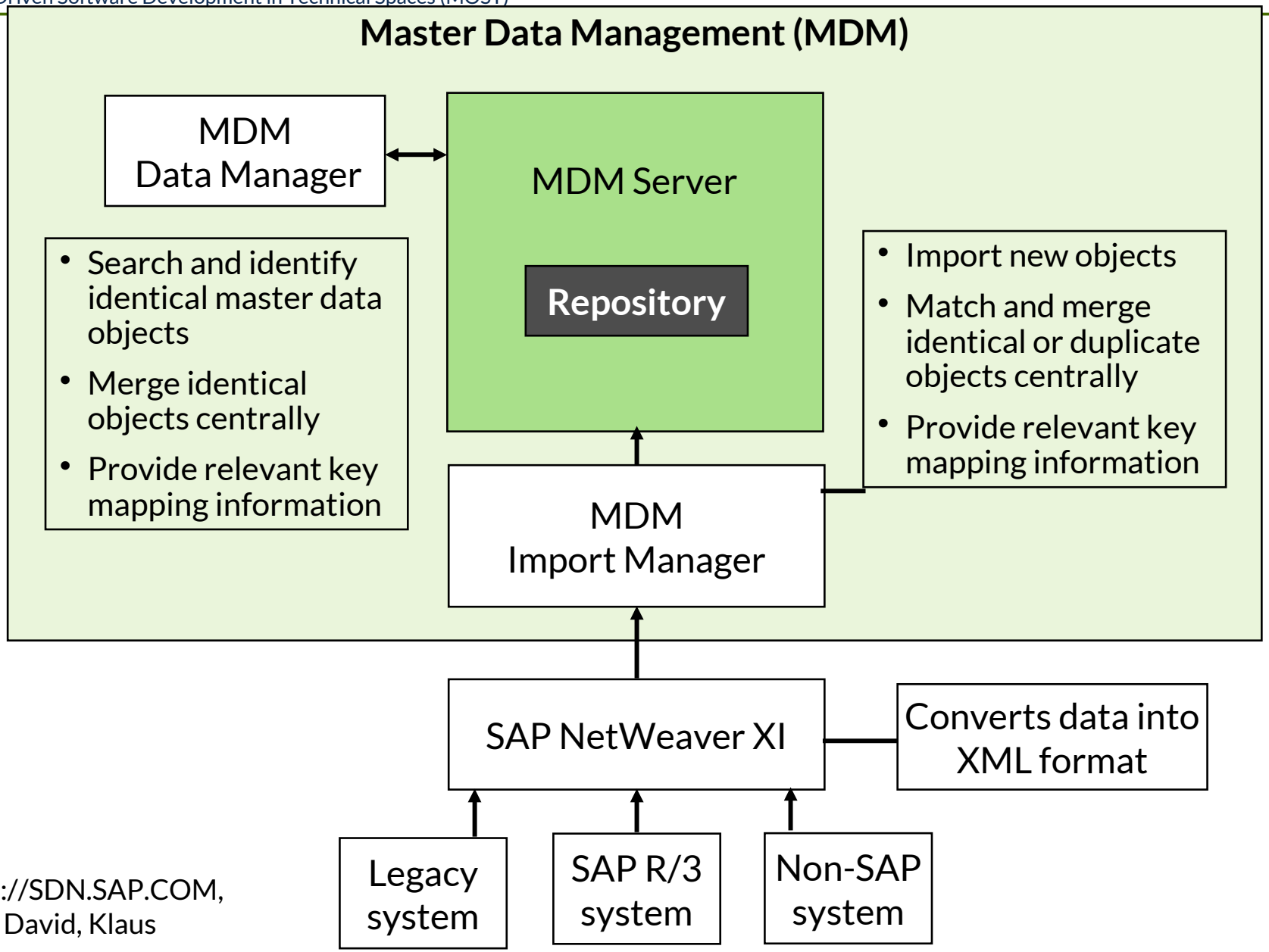
http://www.modelbus.org/modelbus/

42.4.2 Master Data Management (MDM)

MDM

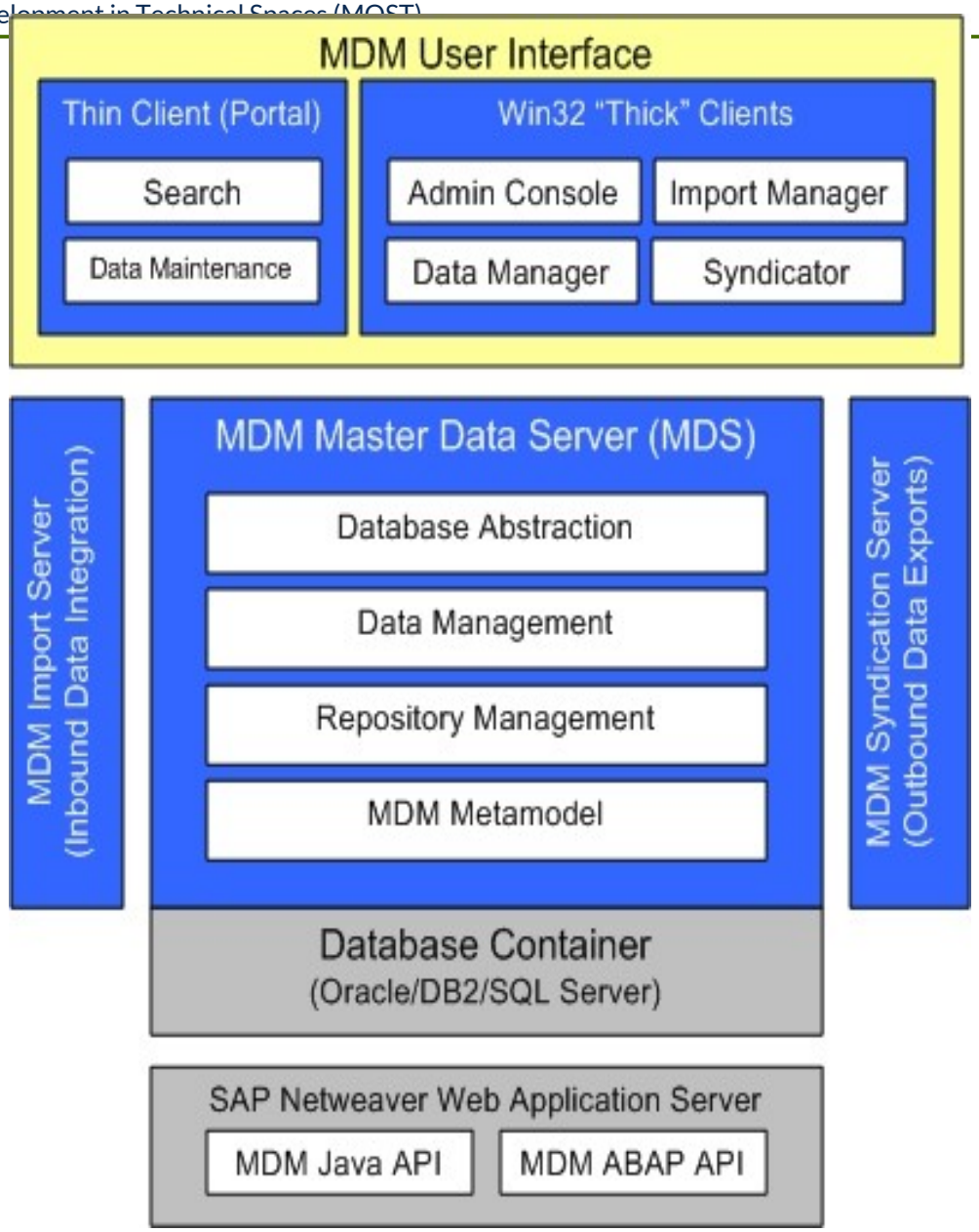
- Company-wide distributed federated repository
  - Consistent (by transactions) or non-consistent
  - http://en.wikipedia.org/wiki/Master_Data_Management
- Often stems of IT mergers of companies and must be “merged” and “consolidated”

Ex.: SAP NetWeaver MDM



Quelle:
URL: http://SDN.SAP.COM,
Artikel of David, Klaus

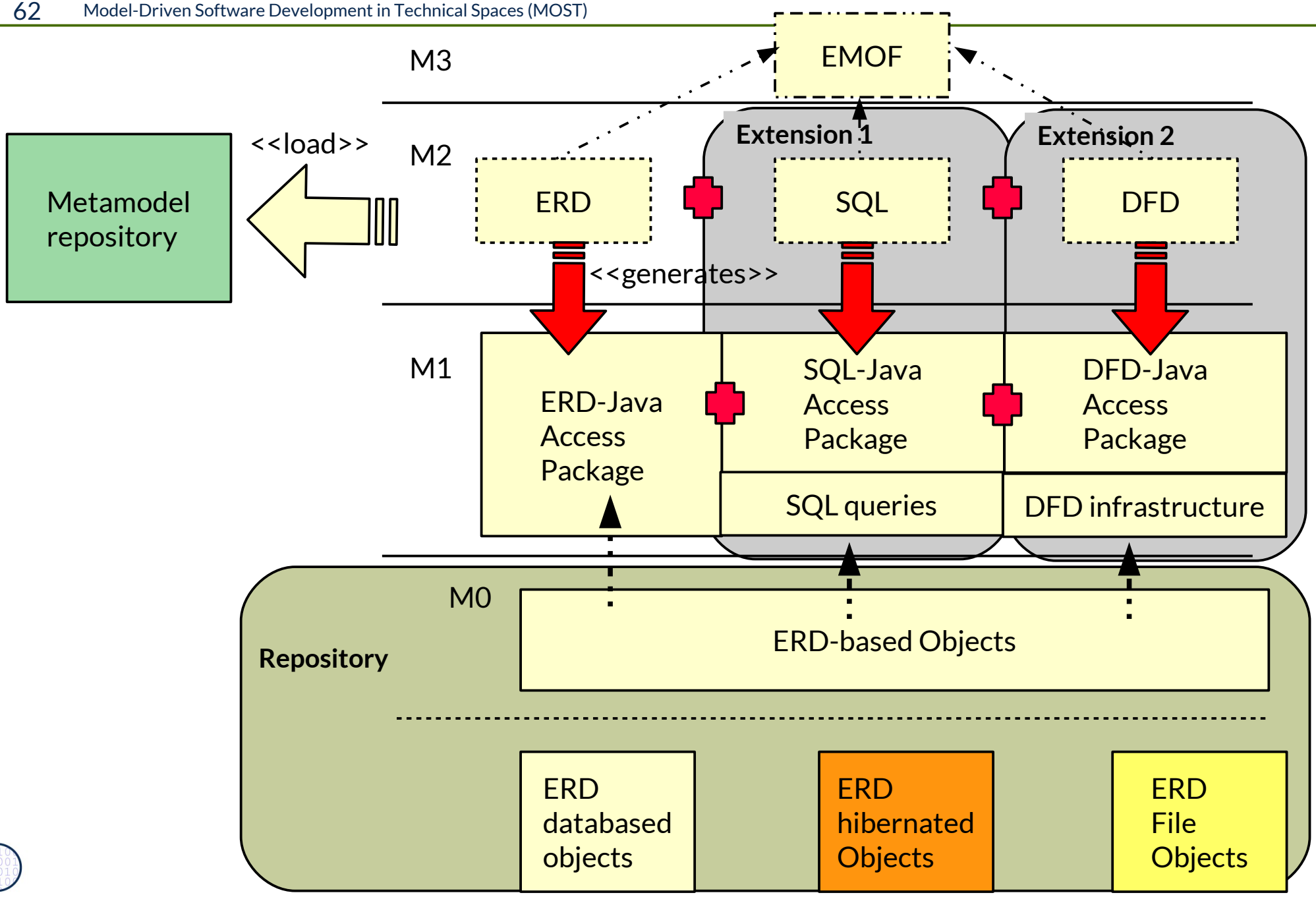
http://searchsap.techtarget.com/resources/SAP-MDM-software



http://searchsap.techtarget.com/generic/0,295582,sid21_gci1232140,00.html

42.5 Extension of Repositories by Metamodel Extension

Extension of Metamodel-Driven Repositories



Why are Metamodel-Driven Repositories Important for MDSD Tools?

Simple extension and composition of typed repositories on M0, M1, M2

- In a metamodel-driven repository, loading of new metamodels extends the access interfaces
  - load the new metamodel
  - generate new typed access interfaces (and code) for access, query, consistency context and roles, e.g., with R.O.P
  - load new code by dynamic class loading
  - run the new code, allocate new extended objects

For an embedded DSL, a repository as extension of the repository of the GPL
For specific techniques, such as distribution or authentification, plugin metamodels can be created

For an IDE or MDSD Tool, repositories are created by composition of
base techniques and base languages

The End

- Explain the structure of a metamodel-driven repository.
- Why can objects of M0, M1 and M2 be stored in such a repository?
- Explain the structure of a MOF-based repository.
- Explain how the interface of an access role (read, write) of the material in the repository can be generated from a metamodel
- Why is an Observer interface for materials important?
- Explain, why TAM pattern language gives a good model for the tool access of a metamodel-driven repository.
- Which roles of a Material Object do you know and what are their tasks?
- What is the “administration aspect”? What is the “infrastructure aspect”?