III. Software Factories

30. Integration of Tools, Exchange Formats, Language Mappings, and Integrated Development Environments

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de
Version 15-0.7, 08.01.16

References

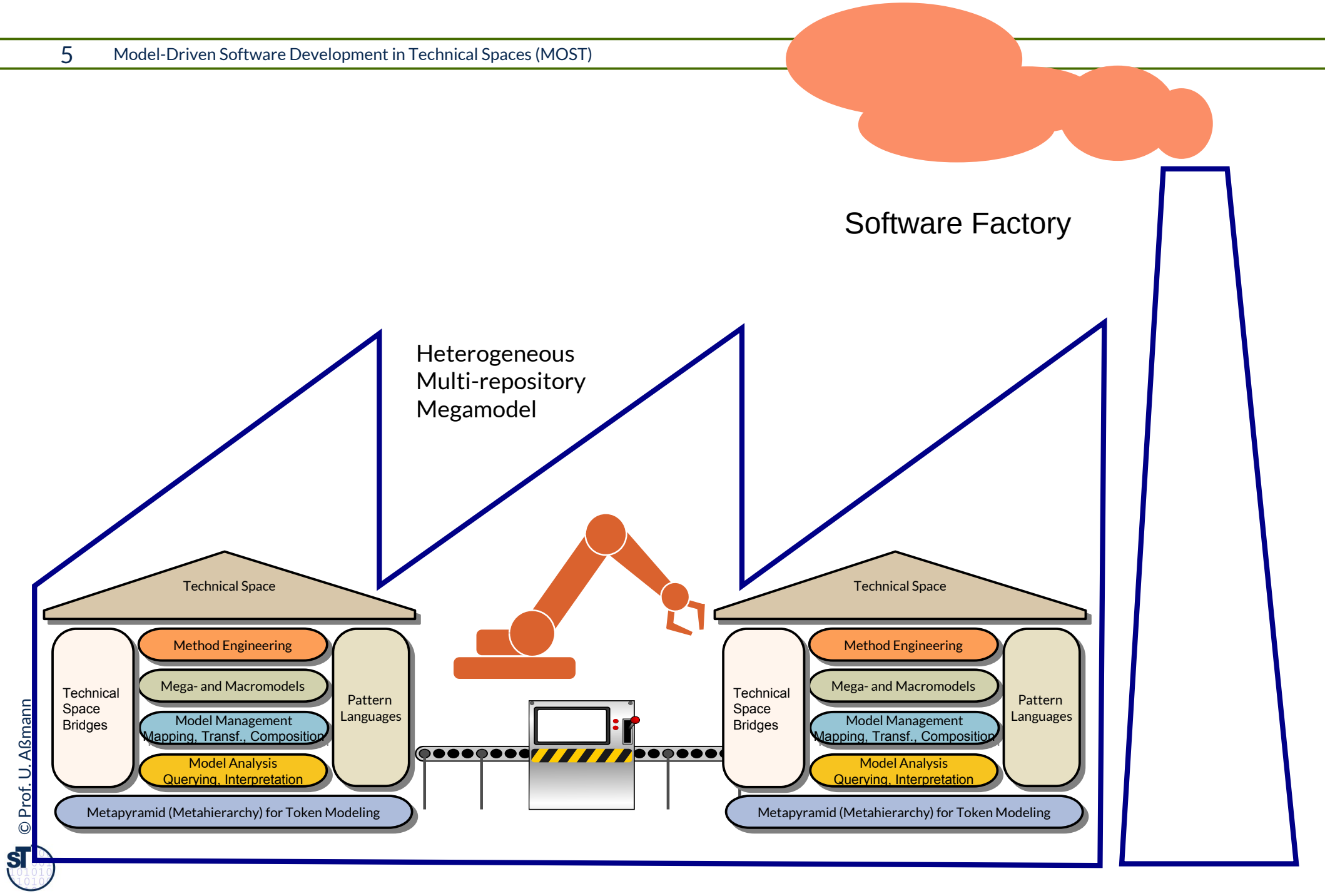
- ECMA, Reference Model for Frameworks of Software Engineering Environments, Technical Report 55, 3rd Edition, Juni 1993
  - http://www.ecma-international.org/publications/files/ECMA-TR/TR-055.pdf
- Richard C. Holt, Andreas Schürr, Susan Elliot Sim, and Andreas Winter. GXL: A graph-based standard exchange format for reengineering. Science of Computer Programming, 60(2):149-170, April 2006.
  - http://www.gupro.de/GXL/Publications/publications.html

The Book of the MOST Project (2008-11)

MOSTPROJECT
http://most-project.eu

30.1 Software Factories with Heterogeneous Repositories

Q13: A Software Factory's Heart: the Multi-TS Megamodel

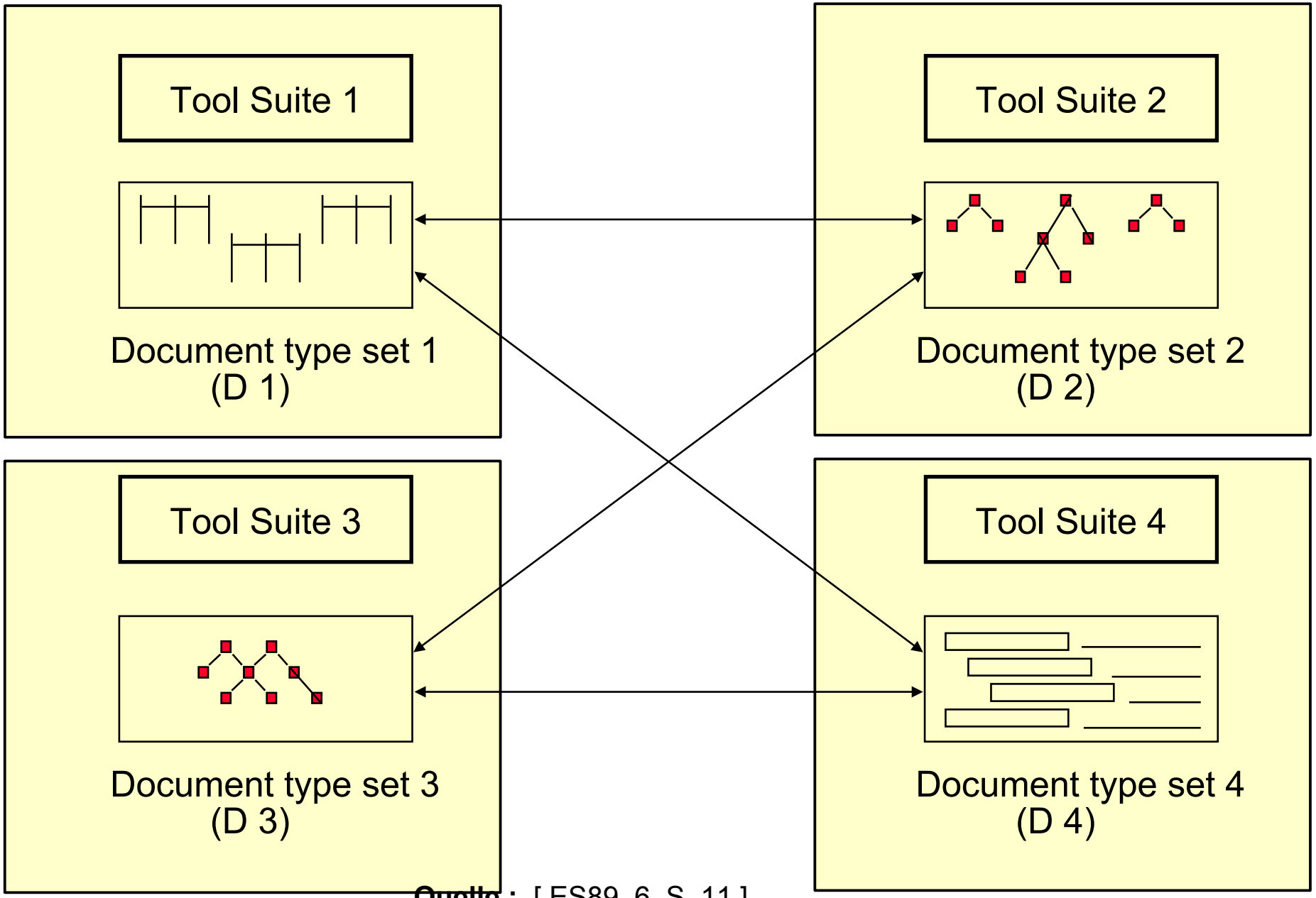


Software Factories

- A **software factory** is a modeling environment for the production of software product lines.
  - The software factory maintains a set of repositories for models in different languages and connects them to one macromodel:
  - **Mono-repository:** all models of the megamodel in one repository, requiring one technical space
  - **Multi-repository:** models in different repositories
  - **Heterogeneous multi-repository:** models in different repositories, with many technical spaces
- Software factories are big integrated MDSD environments coupling many tools and materials
  - A **pure software factory** creates software with software
    - Example: UML tool, SysML tool
  - A **systems software factory** creates a system in soft- and hardware (with software)
    - Example: PreeVision, ASCET, Cadence Silicon Compiler
- Often, multi-repository software factories result from ***data connection*** of already existing tools (tool integration)

30.1.1 Concepts of Tool Integration for Software Factories

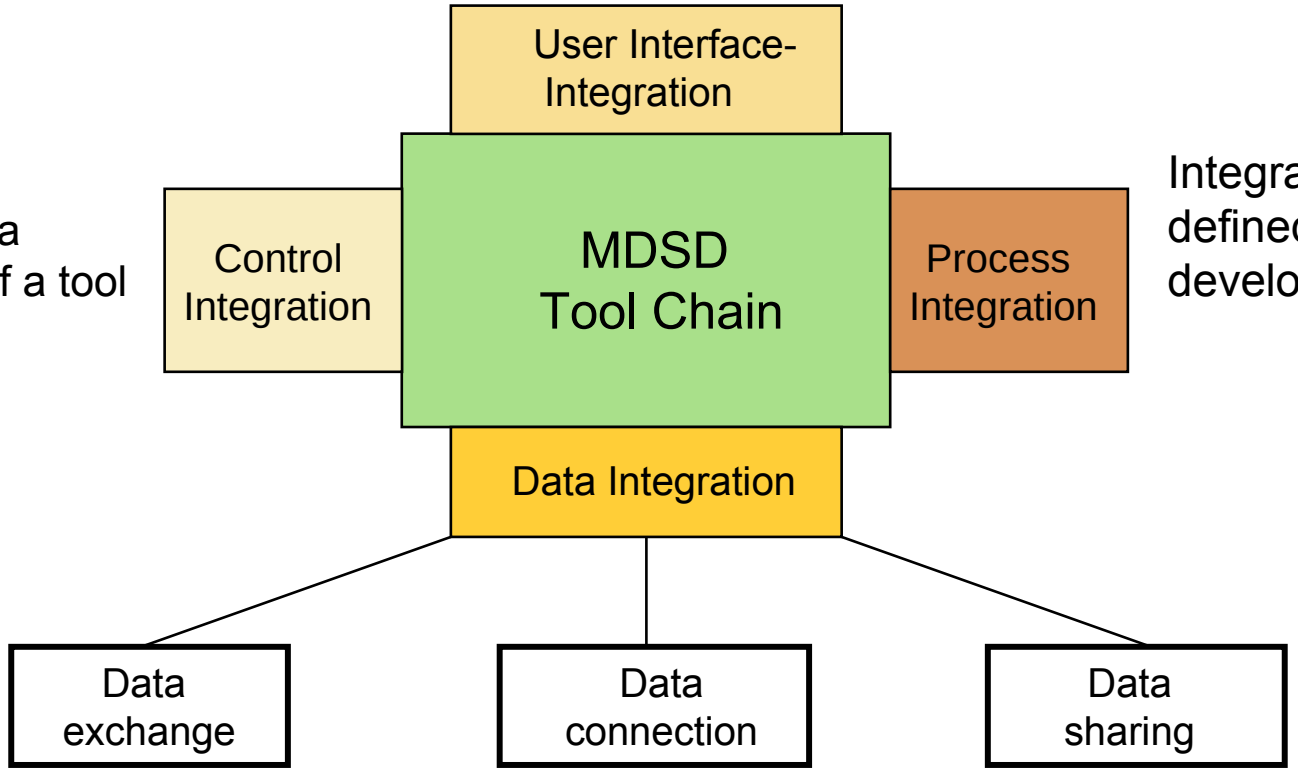
Integration of Tool Suites



Quelle : [ES89, 6, S. 11]

Tool Integration in 4 Dimensions

Embedding into a workflow (e.g., of a tool automaton)



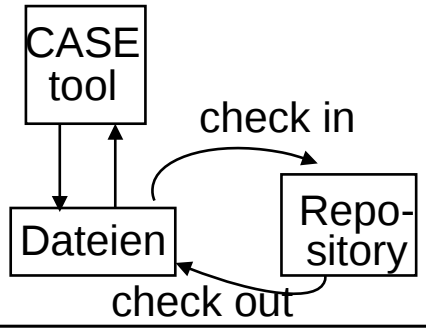
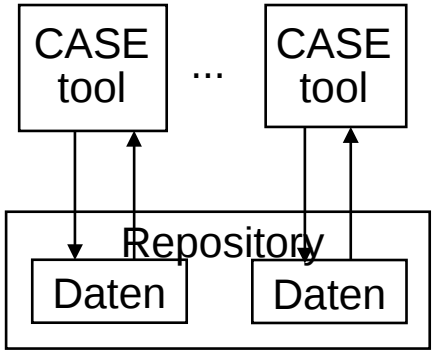
Integration into a defined software development process

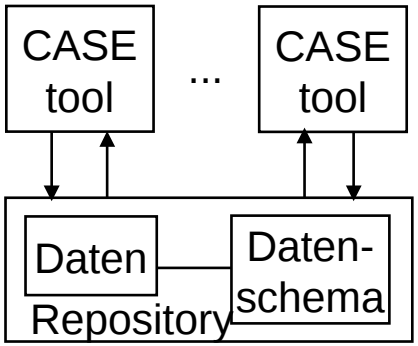
Quelle: [Balzert-II 3, S. 605]

30.2 Data Integration

Levels of Data Integration

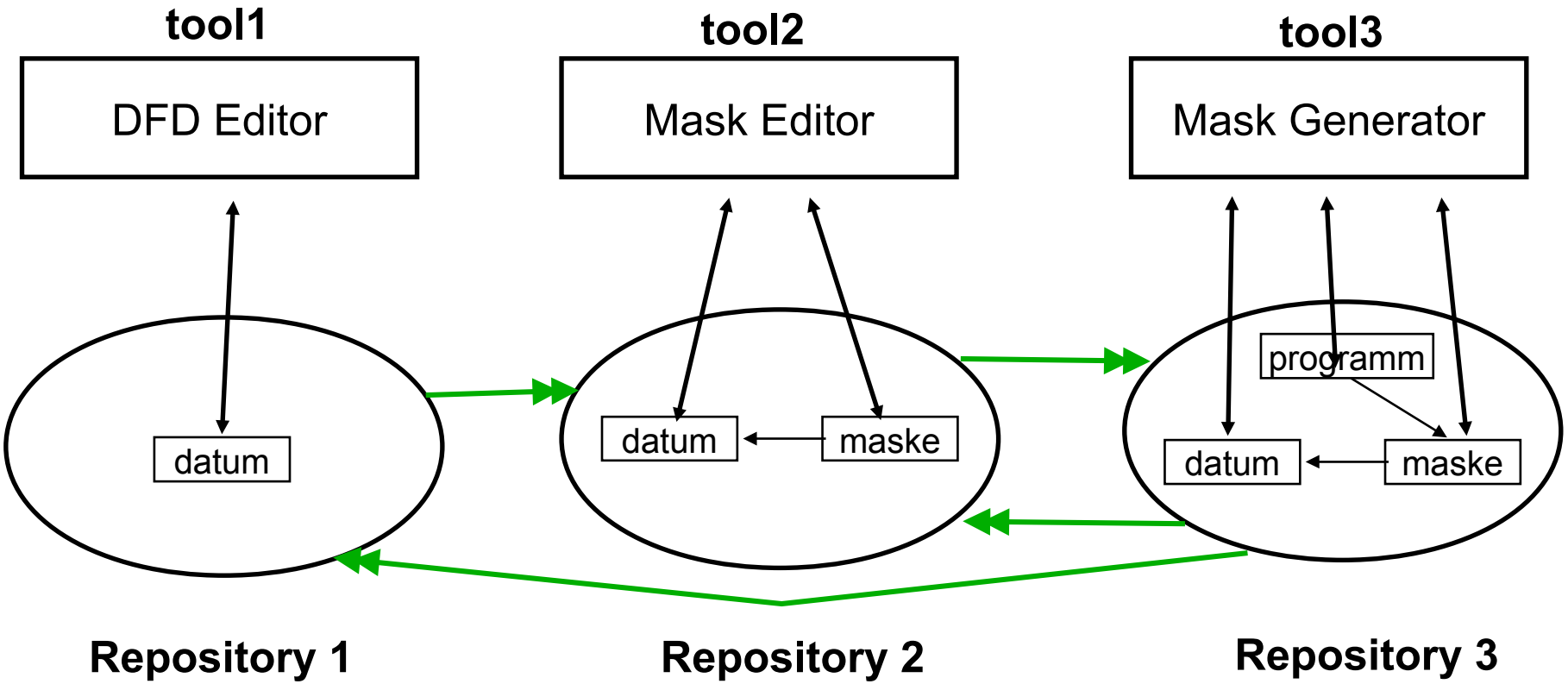
[Balzert-II, S. 608]

Integration Kind	Schema	Characteristics
Black-box-Integration (weak)		• Too works isolated on own data • Repositoriy coordinates data with check in and check outl
Grey-box-Integration (medium)		• Separate repositoriy, but common access interfaces • Access layer to repositories

White-box-Integration Data sharing (strong)		• Definition of uniform data schema (material metamodels) for all tools
		• Integration of data schemata of tools by
		  • Role-based integration
		  • Inheritance-based integration

a) Data Exchange (Simple)

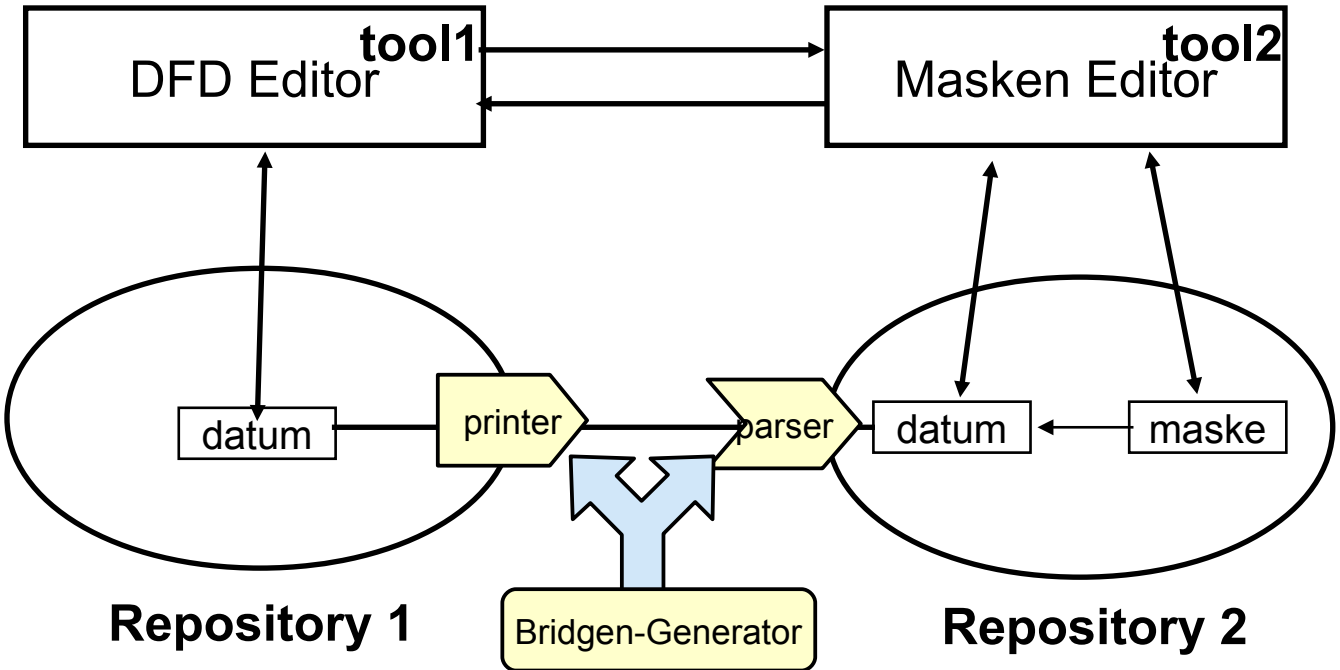
- No common data, high cost for exchange
- Exchange with Streams and Data-Flow Architecture (Example: UNIX shell)
  - Queries filter data, transformations change data
  - Data formats are defined in an exchange language
- Tools are rather independent



Quelle: nach [HMF. S. 196]

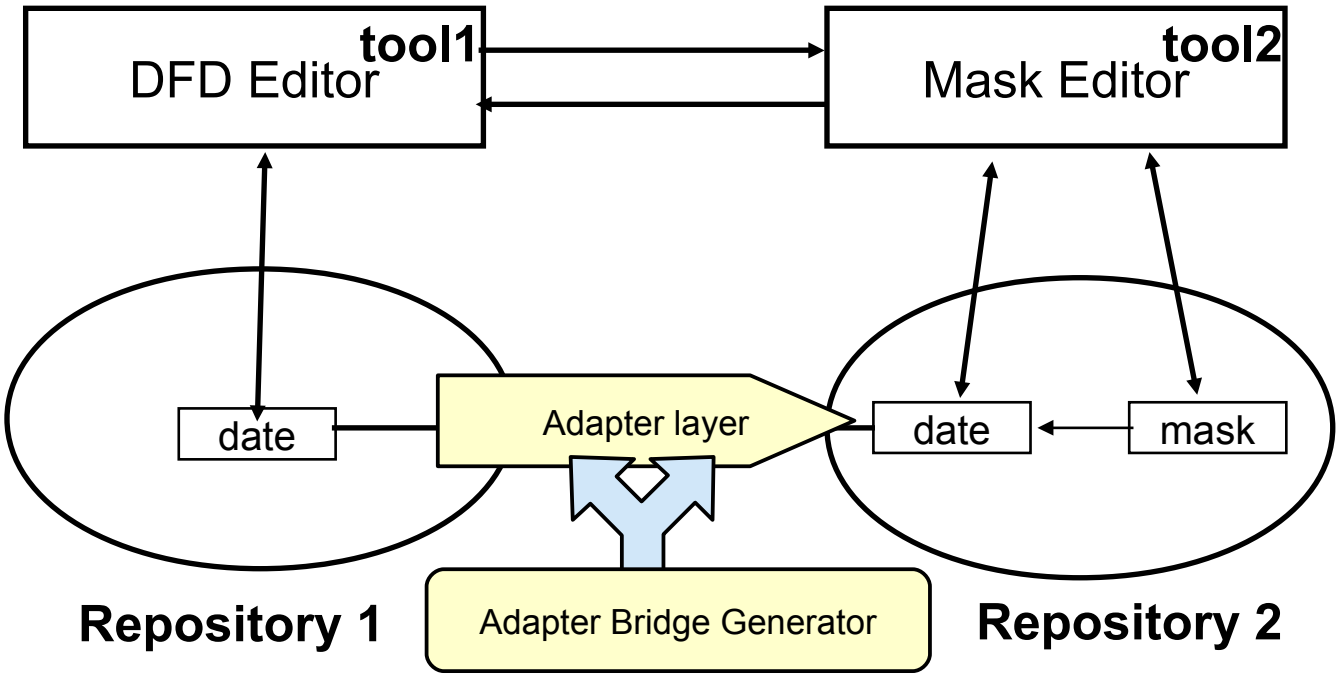
b) Data Connection with Systematic Data Exchange (Transformation Bridges)

- **Data connection (Datenverbindung)** relies on the definition of *mappings between the data schemata (material metamodels)*
- **Automated data exchange** in standardised **exhange formats (Exchange formaten** (z. B. ASN, XMI, JSON, CDIF, XML)
  - Automation relies on mappings between the material metamodels
- **Transformation bridge:** a prettyprinter transforms the internal representation of a repository into an external exchange formant
  - Parser read the exchange format again and transform it into the internal representaiton of the other repository
  - Query and transformation languages filter the data



c) Data Connection with Incremental Data Exchange (Adapter Bridges)

- An **Adapter Bridge** is a layer between two repositories allowing for *incremental access* from one to the other
  - Transformation of data incrementally with each access
  - Direct transformation without exchange format



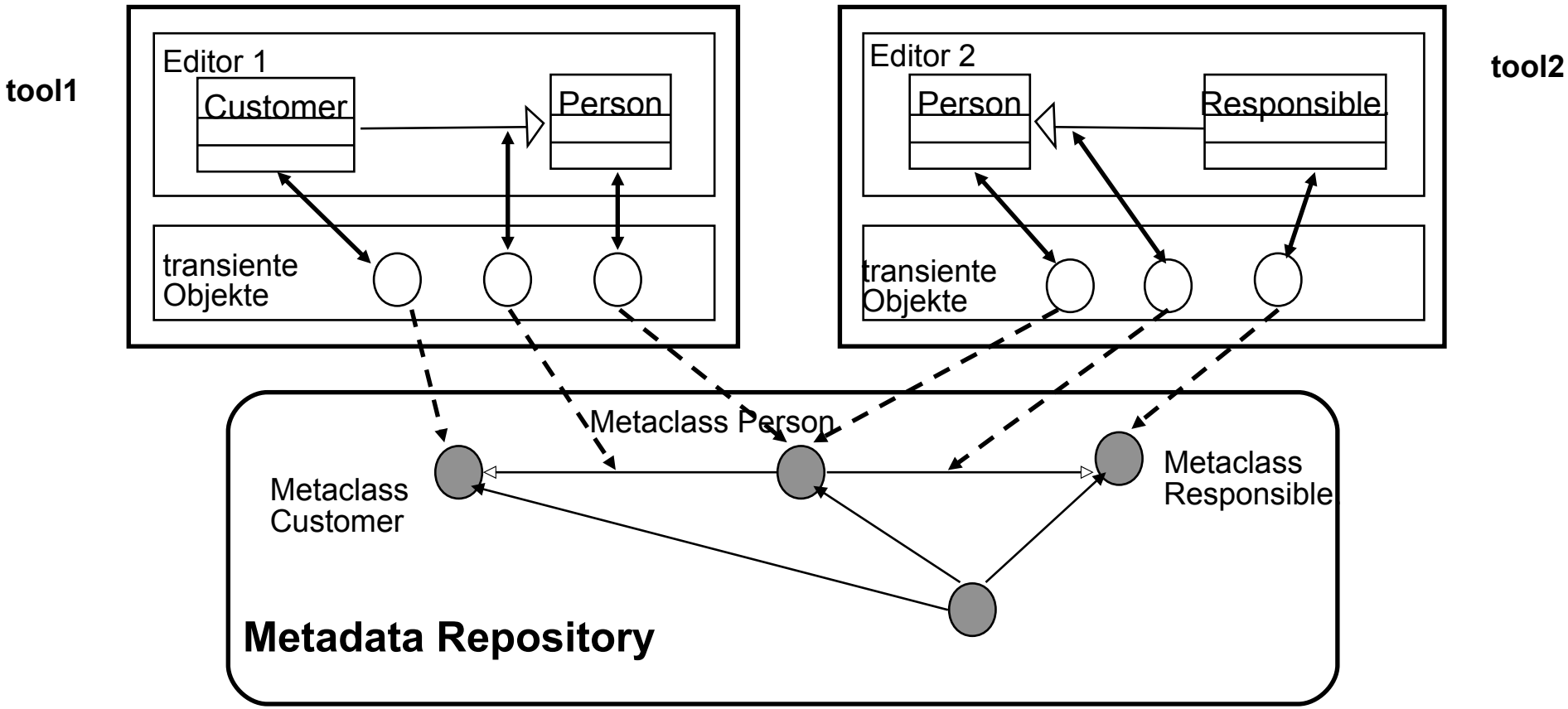
Current Exchange Formats

Exchange format: standardized for the exchange of data between tools, also of different vendors

- Comma-separated values (CSV): simple text-based exchange format for tools on relation, text algebra, and tables (Excel, TeX, ...)
  - No metalanguage but simple table schema http://tools.ietf.org/html/rfc4180
  - http://en.wikipedia.org/wiki/Comma-separated_values
- CASE Tool Data Interchange Format (CDIF) - metalanguage ERD for data definition as well as
  - Data Flow Model, State Event Model, Object Oriented Analysis and Design
  - http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-270.pdf
- XML Metadata Interchange (XMI) for exchange of UML-diagrams in XML-format
  - Meta Object Facility (MOF) as metalanguage
  - http://www.omg.com/technology/documents/formal/xmi.htm
- ASN.1 Standard is a metalanguage based on BNF
  - http://de.wikipedia.org/wiki/Abstract_Syntax_Notation_One
- RDF/RDFS Resource Description Format – Models as graphs, stored in elementary tripels http://www.w3c.org
- GXL Graph Exchange format: Open Source Format for exchange of graphs
  - http://www.gupro.de/GXL/

d) Data Sharing (With Common Repository)

- With **data sharing** (**Datenteilung)** all tools share common data with a uniform schema (material metamodel)
- Metaclass mappings control the integration of the repositories and metamodels



Quelle: Platz, D.,Kelter, U.: Konsistenzerhaltung of Fensterinhalten in SEU; http://pi.informatik.uni-siegen.de

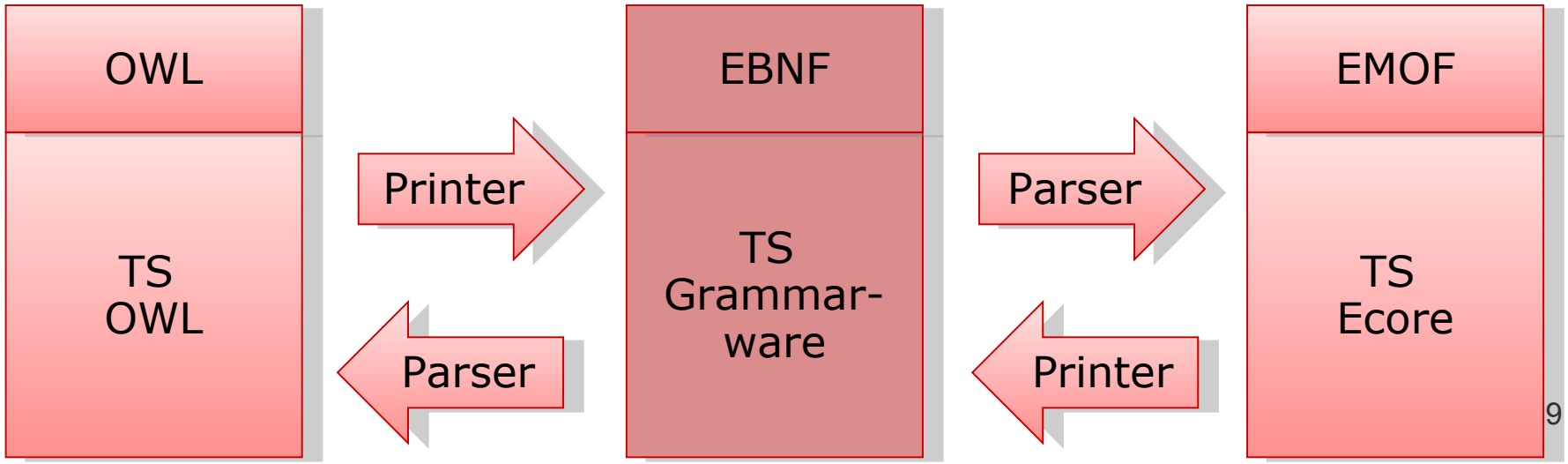
30.3 Data Connection with Exchange Formats

Exchange Formats use Normative Concrete Syntax

- **Data Connection (Datenverbindung)** between repositories relies on a *semantic relationship of the data*
  - Identical M2-Metamodel
  - Metamodels can be mapped isomorphically (1:1) (**language mapping)**
  - Metamodels can be mapped homomorphically (n:m, n:1), e.g., language A is a subset of language B or vice versa
- Exchange format are text-based with a **normative concrete syntax**, because the syntax is normed (standardized) and not user-centered
  - Grammarware, XMLWare are the most popular technical spaces
  - From the language mappings, parsers and prettyprinters are generated

Transformative TS-Bridges with Concrete Syntax

- A **transformative Technical-Space-Bridge** (TS-Bridge, TS bridge) offers
  - An exchange format in normative concrete syntax (e.g., via TS Grammarware)
  - A generation technique for printers and parsers
- Techniques for normative concrete Syntax
  - Text: EMFText: normative concrete syntax for Ecore/EMOF
  - Text: Xtext: normative concrete syntax for Ecore/EMOF
  - CDIF: normative concrete syntax für ERD
  - JSON: text-based normative concrete syntax



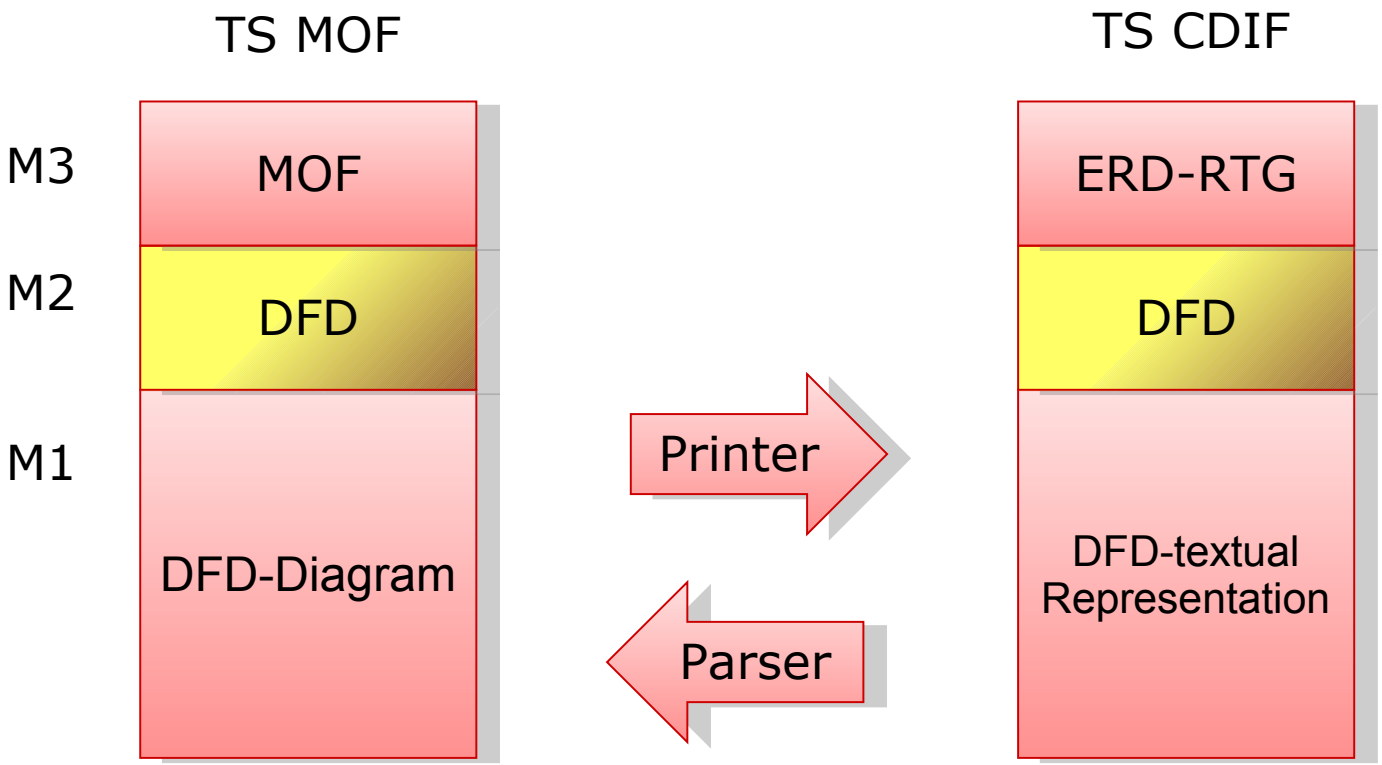
Exchange Format CDIF (CASE Data Interchange Format) Example: Specification of DFD

- CDIF is a historic exchange format for CASE tools based on metalanguage ERD
- The standard CDIF uses RTG for the definition of types, a textual notation of ERD (ERD-Text)
- Example: Specification of DFD in ERD-RDT (keywords in boldface, defined non-terminals in typewriter, used non-terminals in italics):

```
obj_dfd          (dataFlowDiagram dfd_title {dfd_element} )
dfd_title        (dfdTitle @dfd_title_id dfd_title_name )...
dfd_element      dfd_bubble |dfd_store | dfd_term | dfd_tb |
                      dfd_csc | dfd_flow
dfd_bubble       (process  pt pt @process_id process_name
                      inst_num [process_type] )
@process_id      (processID string )
process_name     (processName string )
process_type     (processType string )
dfd_store        (store pt pt store_name inst_num )
store_name       (storeName string )
```

Exchange Format CDIF

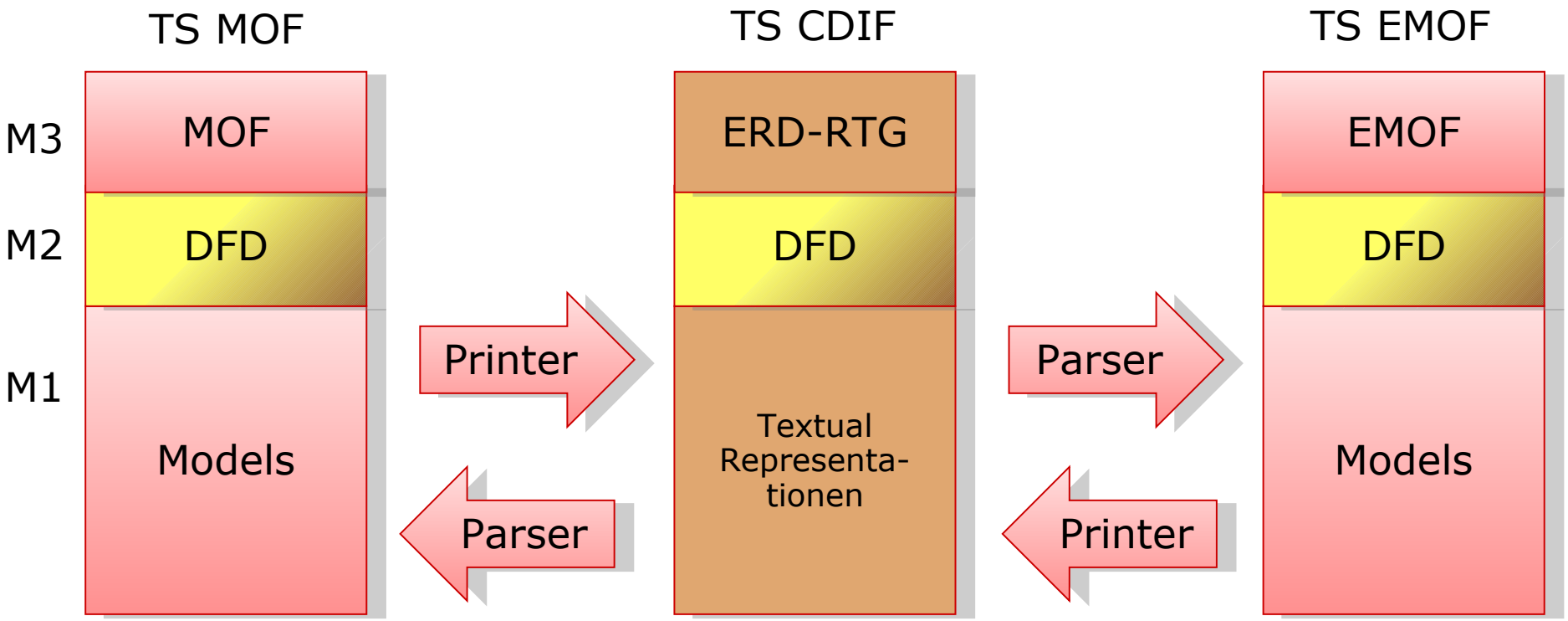
- CDIF defines a normative textual syntax für ERD (ERD-RTG)



30.3.2 Transformation Bridges between Technical Spaces

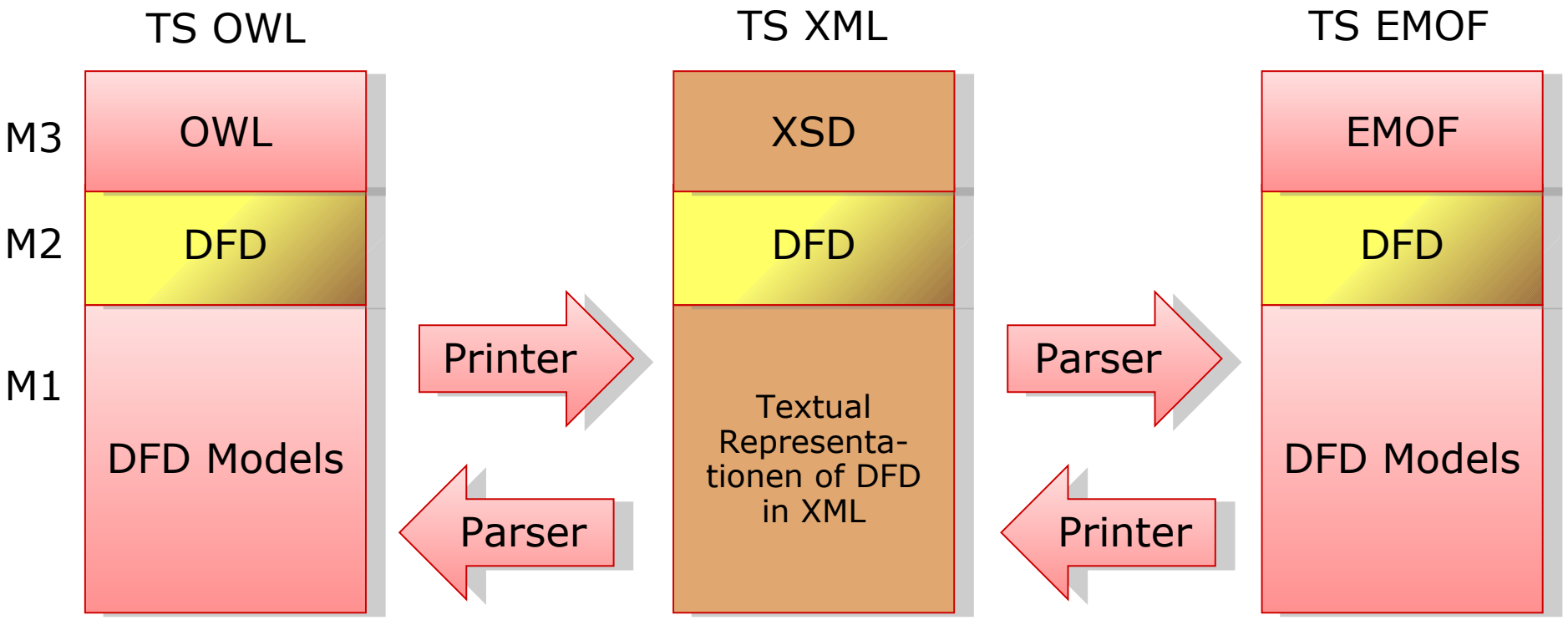
Transformative TS-Bridges via CDIF

- TS-Bridges (via CDIF) generate parser und printer
- Here: TS MOF to EMOF



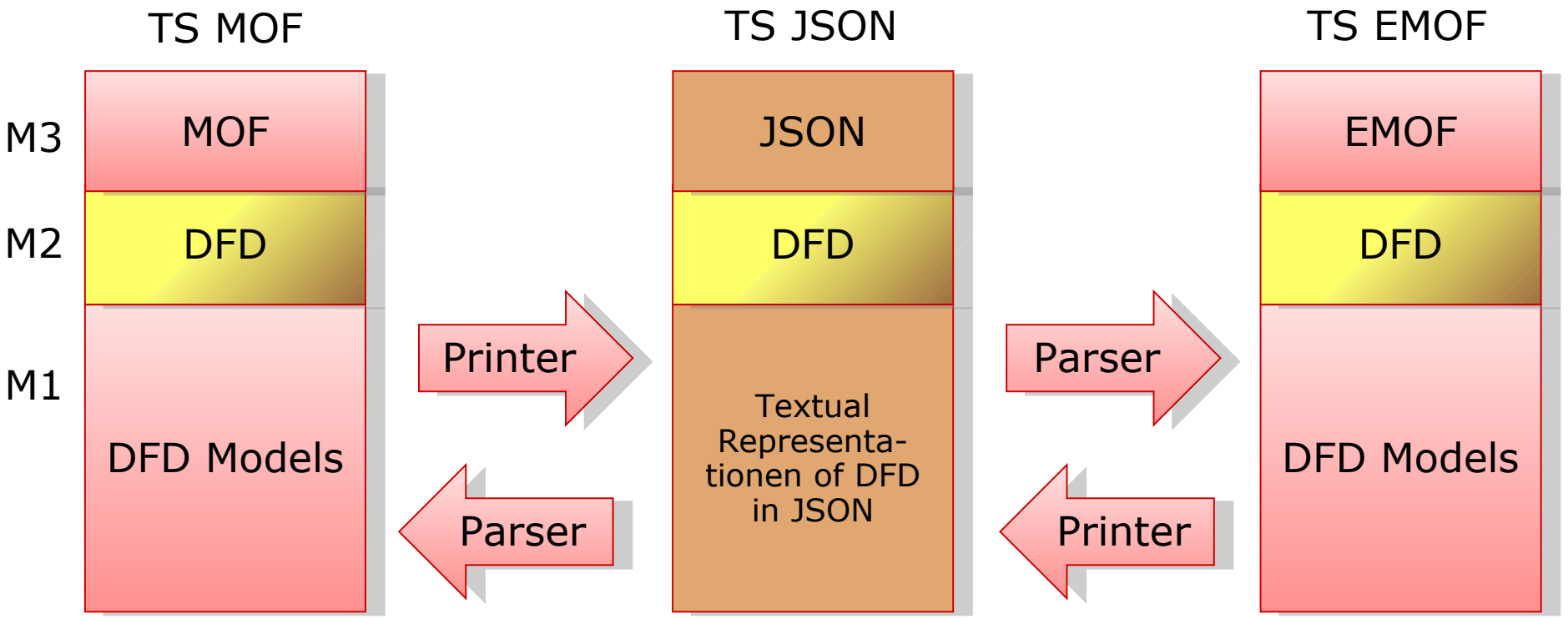
Transformative TS-Bridges via XML

- XML is a normalized concrete syntax
  - Again, because of indeterministic spanning trees, a linearized normalized concrete syntax is possible
- Hoever, good for exchange!



Transformative TS-Bridges with JSON

- JSON (Java Script Object Notation) has established as simple exchange format for record trees (attributed trees) http://www.json.org/ http://www.ietf.org/rfc/rfc4627.txt?number=4627
- Basically, it is like XML, but better readable

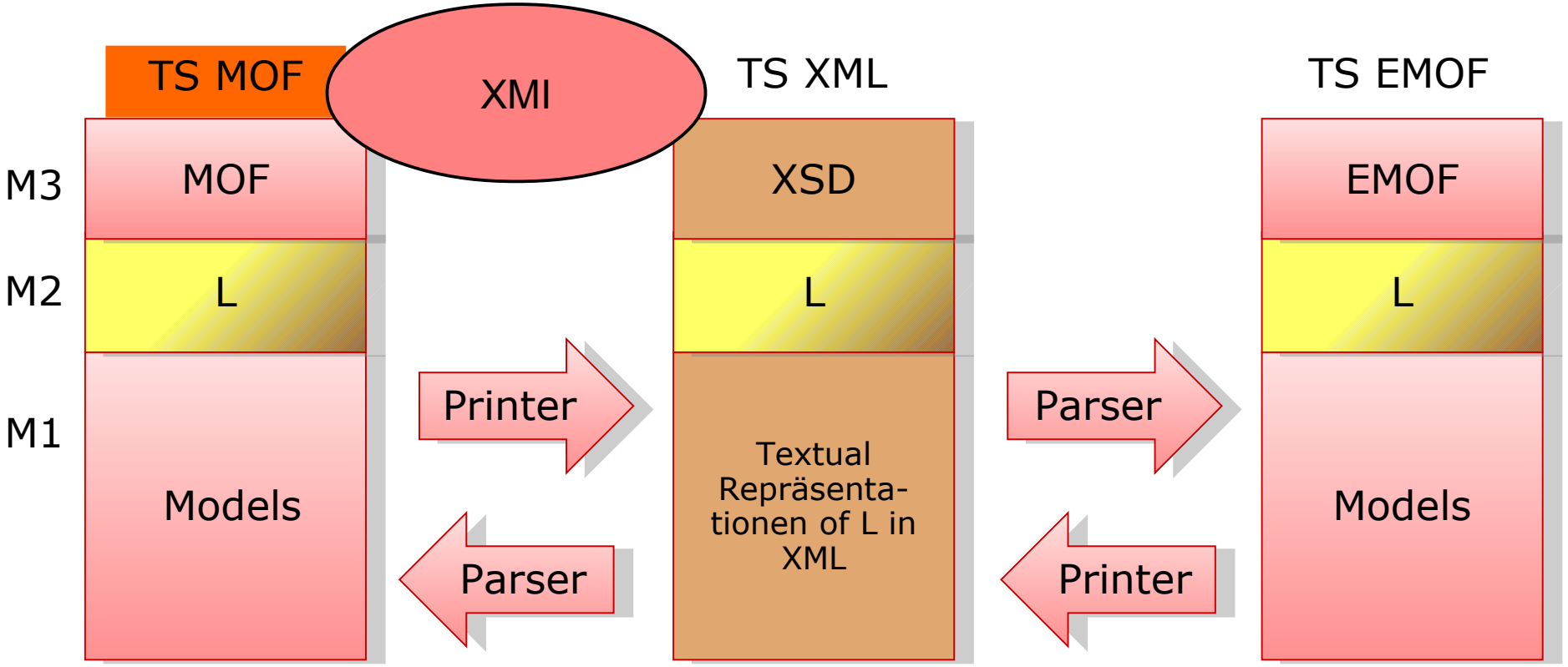


Exchange Format XMI
XML Metadata Interchange Format

- **XML Metadata Interchange (XMI)** is an OMG-Standard for an exchange format
  - Version 2.1 (formal/2005-09-01)
  - Metalanguage MOF
  - generic „Stream“ format
- Problem:
  - Graph-based models must determine a spanning tree (for the XML link tree)
  - There are several spanning trees (Indeterminism)
  - No full compatibility between tools!

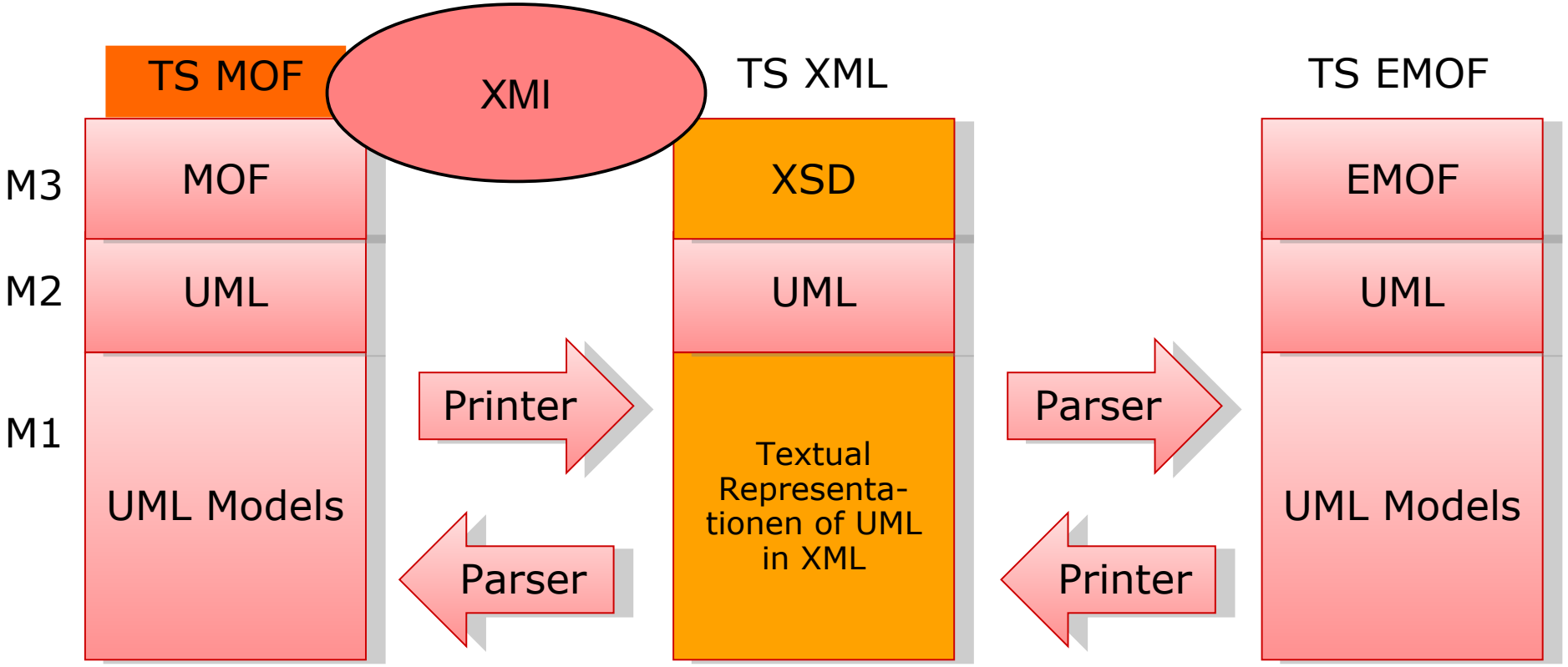
XMI: Transformative TS-Bridge

XMI is a TS-Bridge between the TS MOF and other TS via XSD/XML



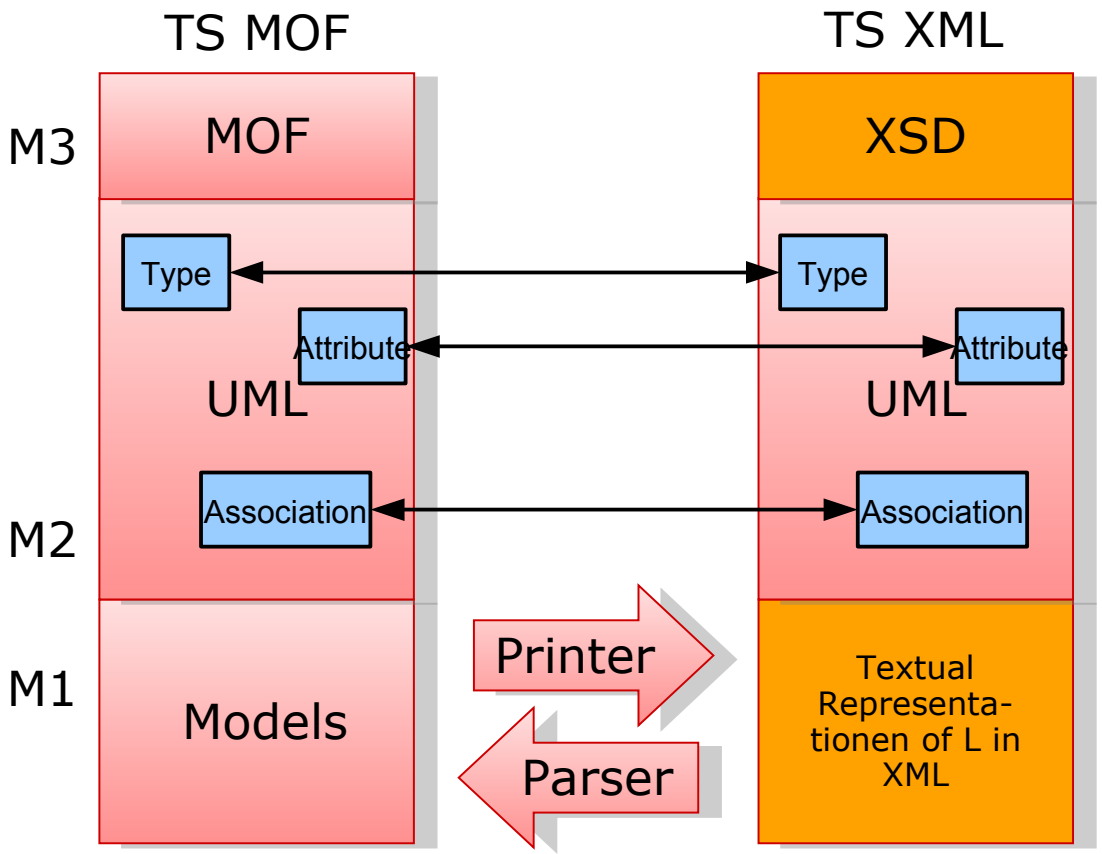
XMI: Transformative TS-Bridge for UML

- Often, XMI is only used for UML. Then, XMI is an UML-Bridge



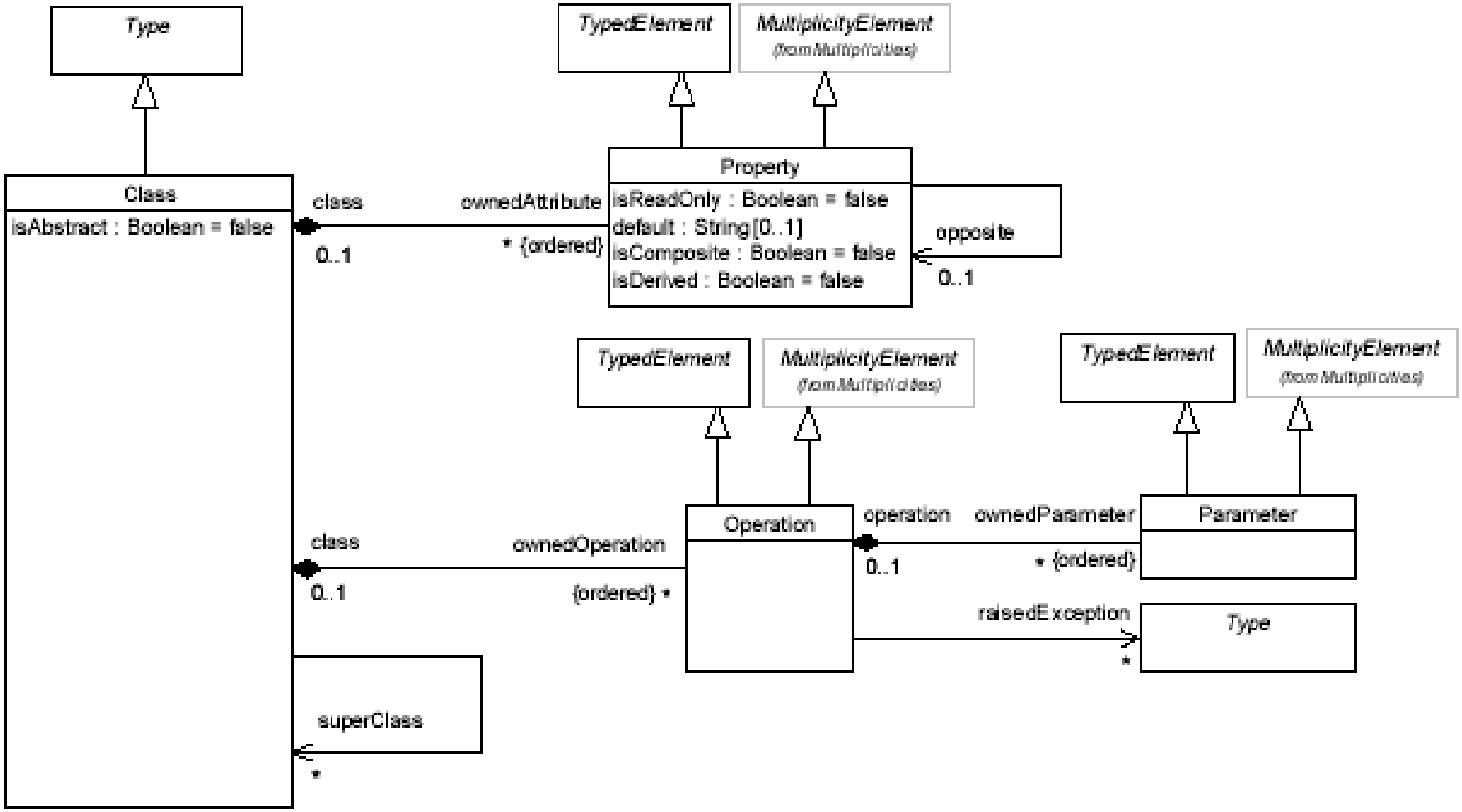
Relation XMI – UML: Language Mappings

- XMI has as a basis the UML metamodel specified both in MOF und XSD
  - Between these metamodel a **bidirectional, isomorphic language mapping** is given
  - Printer and parser are generated from this mapping



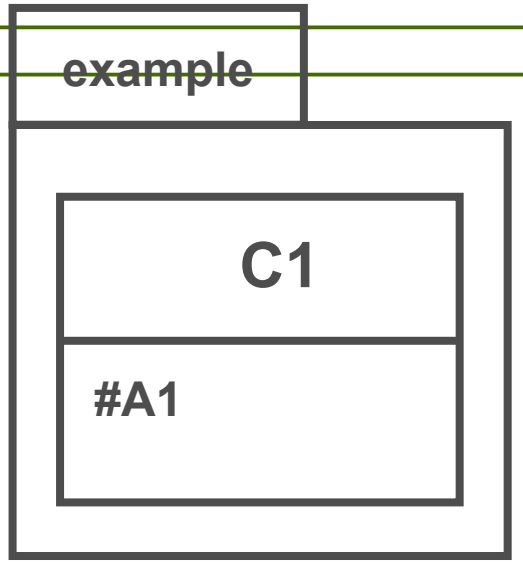
Remember: Classes and Properties in UML-Core

Metamodel for Class and Property



Quelle: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15

Example of an XMI-Instance: Coding of an UML-Class



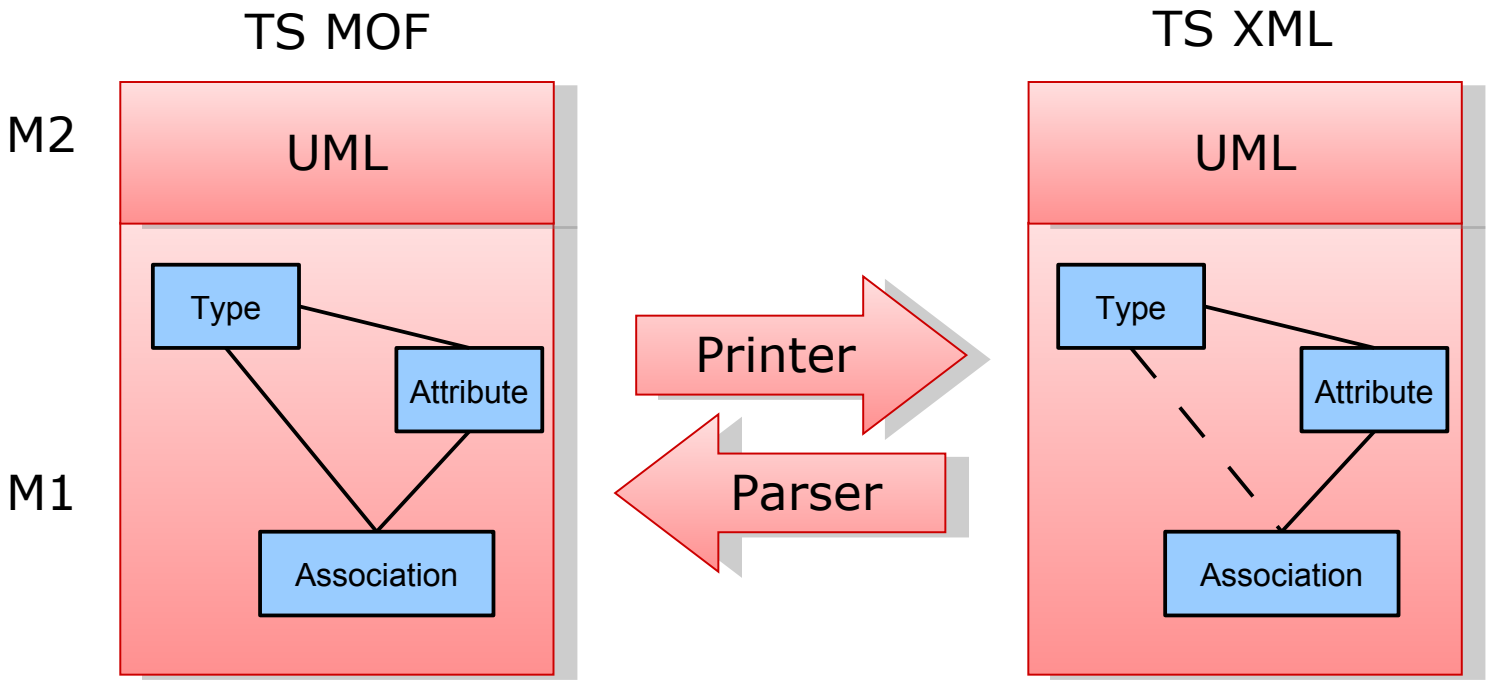

```
<?xml version = „2.0"?>
<!DOCTYPE XMI SYSTEM "uml.dtd">
<XMI xmi.version=„2.0">
<XMI.Header>
      <XMI.Metamodel name=„UML“ href=„UML.xml“/>
      <XMI.Model name=„example“ href=„example.xml“/>
</XMI.Header>
<XMI.Content>
<Core.Basic.NamedElement.name>example</Core.Basic.NamedElement.name>
   <Core.Basic.Class>
    <Core.Basic.NamedElement.name>C1</Core.Basic.NamedElement.name>
      <Core.Basic.feature>
        <Core.Basic.Property>
         <Core.Basic.NamedElement.name>A1</Core.Basic.NamedElement.name>
         <Core.Sasic.NamedElement.visibility xmi.value=“protected"/>
        </Core.Basic.Property>
        [<Core.Basic.Operation> ... </Core.Basic.Operation>]
      </Core.Basic.feature>
   </Core.Basic.Class>
</XMI.Content>
</XMI>
```

UML Metaclasses

(adopted from: www.jeckle.de/xmi_ex1.htm)

Problem: Normative Spanning Tree for Graph Models

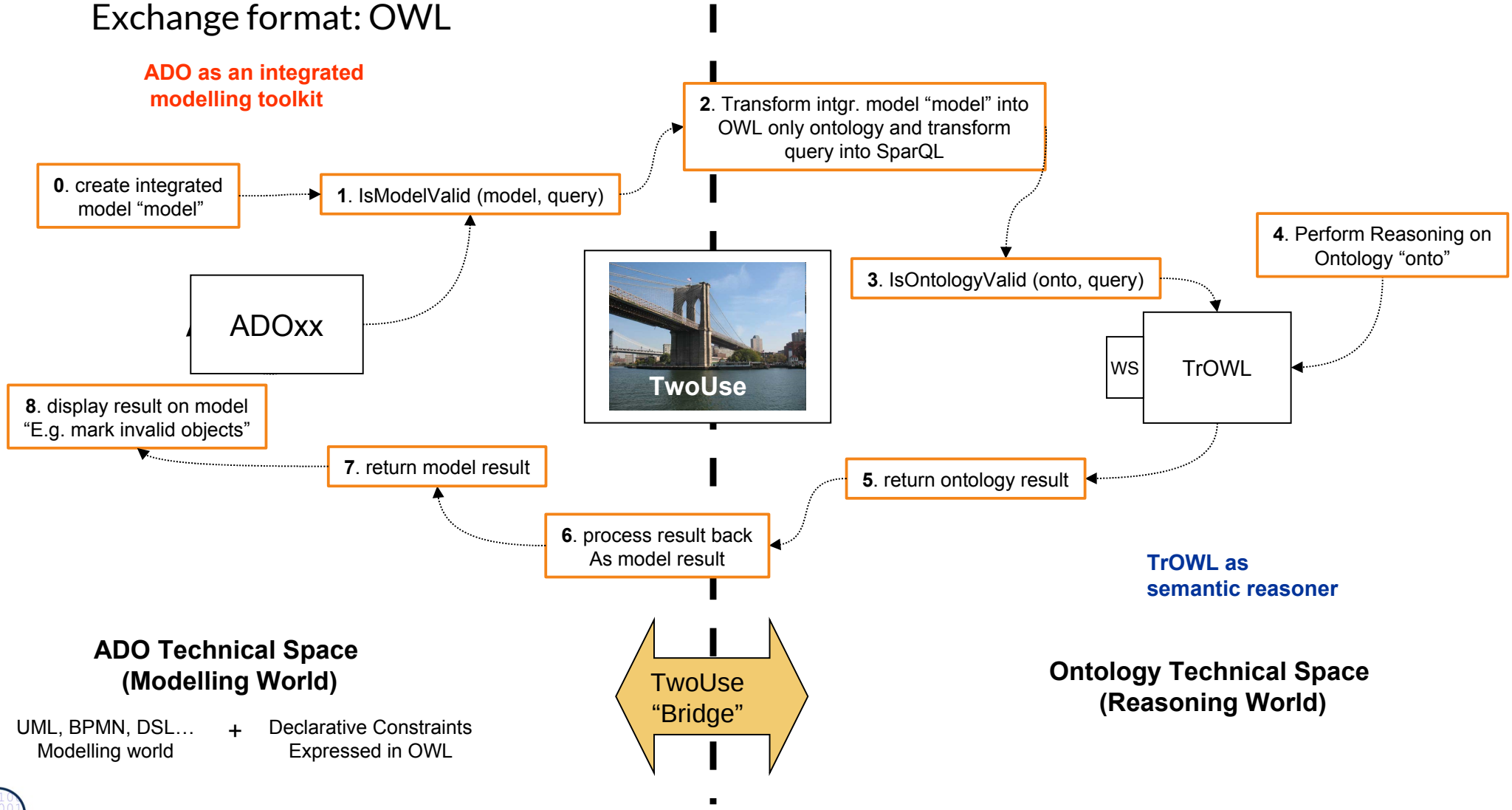
- UML and. MOF are graph-based, while XML is based on link trees
- XML must resolve some links as name references (i.e., revert name analysis)
  - For the UML or MOF-Model, a spanning tree must be found (e.g., along aggregations)
  - All links not in this spannign tree are represented by name references (secondary links), but **not in the XML tree**
- Spanning trees are indeterministic. Thus different linearizations for one graph model possible



Example: Transformation Bridge Between ADO und OWL

TwoUse (U Koblenz) is a transformation bridge between TS ADO (BOC Wien) and TrOWL (OWL, Uni Aberdeen)

Exchange format: OWL



What did we learn?

- Tools can be integrated via
  - Repository integration (data sharing)
  - Stream-based exchange
  - Datenfluss-gesteuerte
- Software Factories (Software MDSD tools, System Engineering tools) are composed from simpler modeling tools *and their materials*
- This notion of “software factory” is due to Prof. Hartmut Fritzsche.

The End

- Why is data sharing the fastest way to integrate tools?
- Explain different forms of language mappings. What is mapped to each other?
  - What is an isomorphic mapping? A homomorphic mapping?
- Compare JSON with XML. What is different, what is similar?
- Explain, why a tree-based exchange format such as XML, has advantages over EBNF based exchange formats.
- Explain, why XML is not perfect as exchange format.

30.A.1 Beispiel einer Referenzarchitektur für Werkzeug-Umgebungen: Das ECMA Referenzmodell für SEU

.. Der ECMA-Toaster

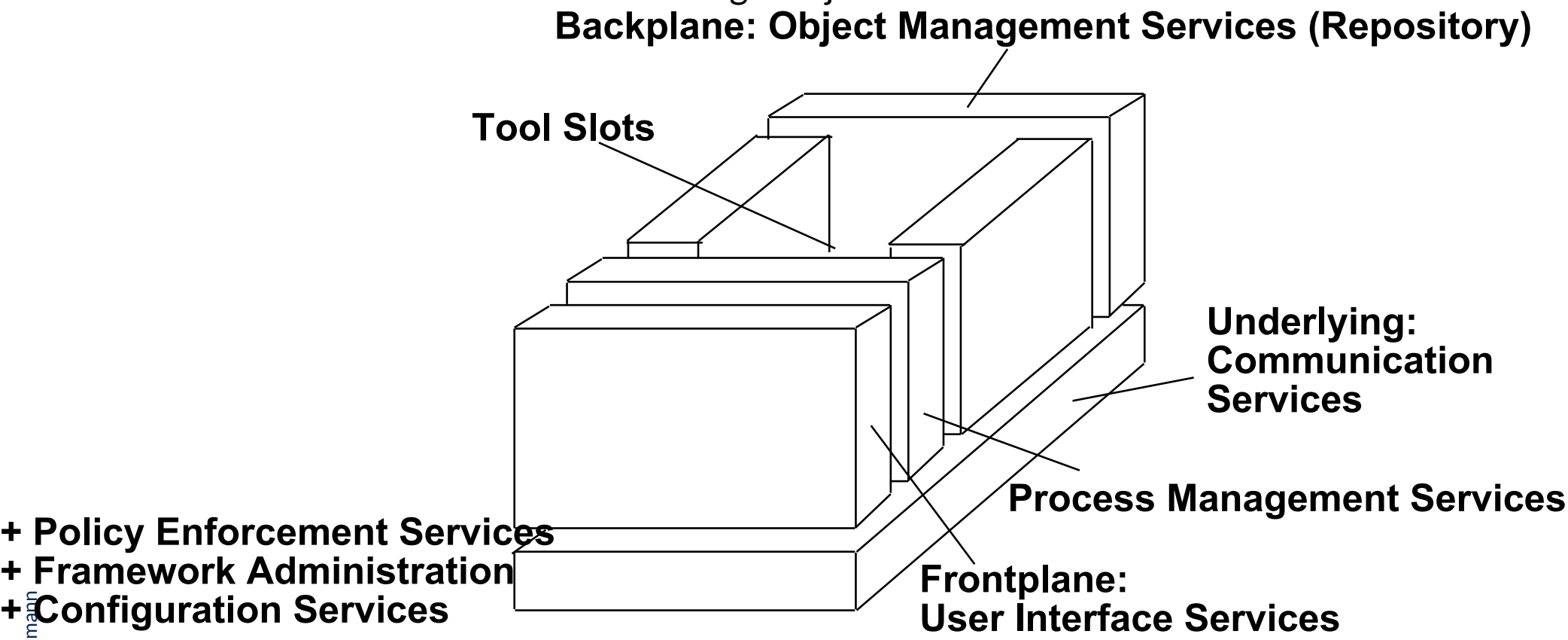
Standardisierungsorganisation
European Computer Manufacturing Association (ECMA)

- Weltweite Normierung der Informationstechnologie und Nachrichtentechnik
  - Mehr als 365 ECMA-Standards
  - 2/3 sind als internationale Standards und/oder technische Reports angenommen worden.
- Ziele:
  - Zusammenarbeit mit nationalen, europäischen und internationalen Normierungsorganisationen über die Standardisierung of Kommunikationstechnologien (ICT) und Verbraucherelektronik (CER).
  - korrekten Gebrauch of Standards anregen und kontrollieren.
  - Veröffentlichung of Standards und technischer Reports, Unterstützung ihrer Verbreitung auch in elektronischer Form
- ECMA hat u. a. folgende Technischen Ausschüsse:
  - TC 32: Kommunikation, Netze und Systemverbindungen
  - TC 39: Programmieren und Script-Sprachen
  - TC 43: Universal 3D (U3D)
  - TC 12: Sicherheit

Quelle: http://www.ecma-international.org/

ECMA-Referenzmodell („ECMA Toaster“)

- Der ECMA Toaster nutzt eine Service-orientierte Architektur (SOA), kann also verteilt sein
- Seine Dienste sind mehr oder weniger in jeder SEU vorhanden



Quelle: ECMA, Reference Model for Frameworks of Software Engineering Environments, Technical Report 55, 3rd Edition, Juni 1993
http://www.ecma-international.org/publications/files/ECMA-TR/TR-055.pdf

Sichten auf Dienste des ECMA-Referenzmodells

externe Sicht	beWritet die externe Nutzung des Dienstes durch andere Dienste bzw. durch Werkzeuge od. den Nutzer
konzeptionelle Sicht	beWritet die Semantik (Funktionalität), ohne Implementierung oder Verfügbarkeit für den Nutzer zu beachten
interne Sicht	beWritet die spezifische implementation (Betriebssystem, andere Tools) für die Dienstausführung
Sicht auf Operationen	führt die Menge of Operationen eines Dienstes auf, die zur Erreichung der Funktionalität (konzeptionelle Sicht) benötigt wird
Sicht auf Typen	beWritet das Datenmodell des Dienstes einschließlich der Informationen über dieses Datenmodell (Metamodell)
Sicht auf Regeln	beWritet Regelmenge, die mögliche Menge der Operationen (Sicht auf Operat.) und annehmbare Zustände der Daten definiert
Sicht auf Dienst-zu-Dienst-Beziehungen	anhand typischer Beispiele wird gezeigt, wie ein Dienst mit einem anderen kommunizieren kann

ECMA Benutzungsschnittstelle
USER INTERFACE SERVICES

ECMA stellt eine Reihe of UI-Diensten (services, Schnittstellen) zur Verfügung, die zur Gewährleistung der Benutzungsschnittstellen-Integration und der konsistenten Bedienung of Anwendungen benötigt werden.

- **User Interface Metadata Service** dient der Definition, Steuerung und Handhabung of Schemata zur Unterstützung der Benutzungsschnittstelle
- **Session Service** gewährleistet volle Funktion, unabhängig of Nutzer oder Hardwareumgebung
- **Security Service** gewährleistet Sicherheitsanforderungen, wie Nutzerauthentifikation, Dunkelsteuerung unbenutzbarer Funktionen u. a.
- **Profile Service** gestattet mögliche Veränderungen, wie z. B. Systemeinstellungen (Farbe), Menge zu verwendender Werkzeuge u.a.
- **User Interface Name and Location Service** stellt fest, wer sich wo zum System Zutritt verschafft hat (logging in)
- **Internationalization Service** stellt nationale Besonderheiten (z. B. Zeichensätze, Datumsformate) zum Zugriff auf das Rechnersystem bereit und gewährleistet ihre Konvertierbarkeit zwischen unterschiedlichen Ländern.

ECMA Prozessverwaltung
Process Management Services

Die **Process Management Services** definieren und organisieren die Ausführung aller Werkzeuge:

- **Process Definition Service** definiert aus den im Repository gespeicherten Projektdaten die Bedingungen zur Ausführung neuer Aktivitäten
- **Process Control Service** steuert Prozesse im allgemeinen auf dem Niveau eines bestimmten Vorgehensmodells zur Beeinflussung anderer Prozesse, wodurch der Nutzer entsprechend seiner role unterstützt wird
- **Process Enactment Service** unterstützt und bietet Möglichkeiten der Steuerung vorher definierter Aktivitäten (Analyse-, Hilfe-, Simulationsfunktionen)
- **Process Visibility and Scoping Service** legt zum Zwecke der Kommunikation und Koordination Sichtbarkeit, Zeitpunkt und Ort of Aktivitätsteilen für andere Aktivitäten fest
- **Process State Service** sammelt und wertet Ereignisse of Aktivitäten während ihrer Ausführung aus, die für die Koordination und spätere Entscheidungsplanung anderer Projektaktivitäten notwendig sind
- **Process Ressource Management Service** verwaltet das Festlegen of Ressourcen zur Ausführung definierter Prozesse für Werkzeuge und Nutzer

ECMA Werkzeugdienste (Tool Services)

- Werkzeuge können in den ECMA Toaster eingesteckt werden bzw. ausgetauscht werden
  - Die gesamte Toolmenge soll nach außen hin durch eine **einzige Schnittstelle** repräsentiert werden.
  - Die Menge der Tools soll den **Softwareentwicklungsprozess vollständig abdecken.**
- Die Werkzeuge kommunizieren über den *Communication Service* oder *Object Management Service*
- Wenn Werkzeuge in die SEU **integriert** werden, ist zu prüfen, ob sie Frameworkdienste bieten.
  - Wenn ja, ist zu entscheiden, diese Dienste weiterhin separat zu ermöglichen oder doch auf die Dienste des SEU-Frameworks überzugehen.
- Um für alle Werkzeuge ein **gleiches Erscheinungsbild** zu erhalten, müssen Basisdienste und Dienste eines SEU-Framework nach standardisierten Vorschriften realisiert werden.

ECMA Repository (Repositorium)
Object Management Services

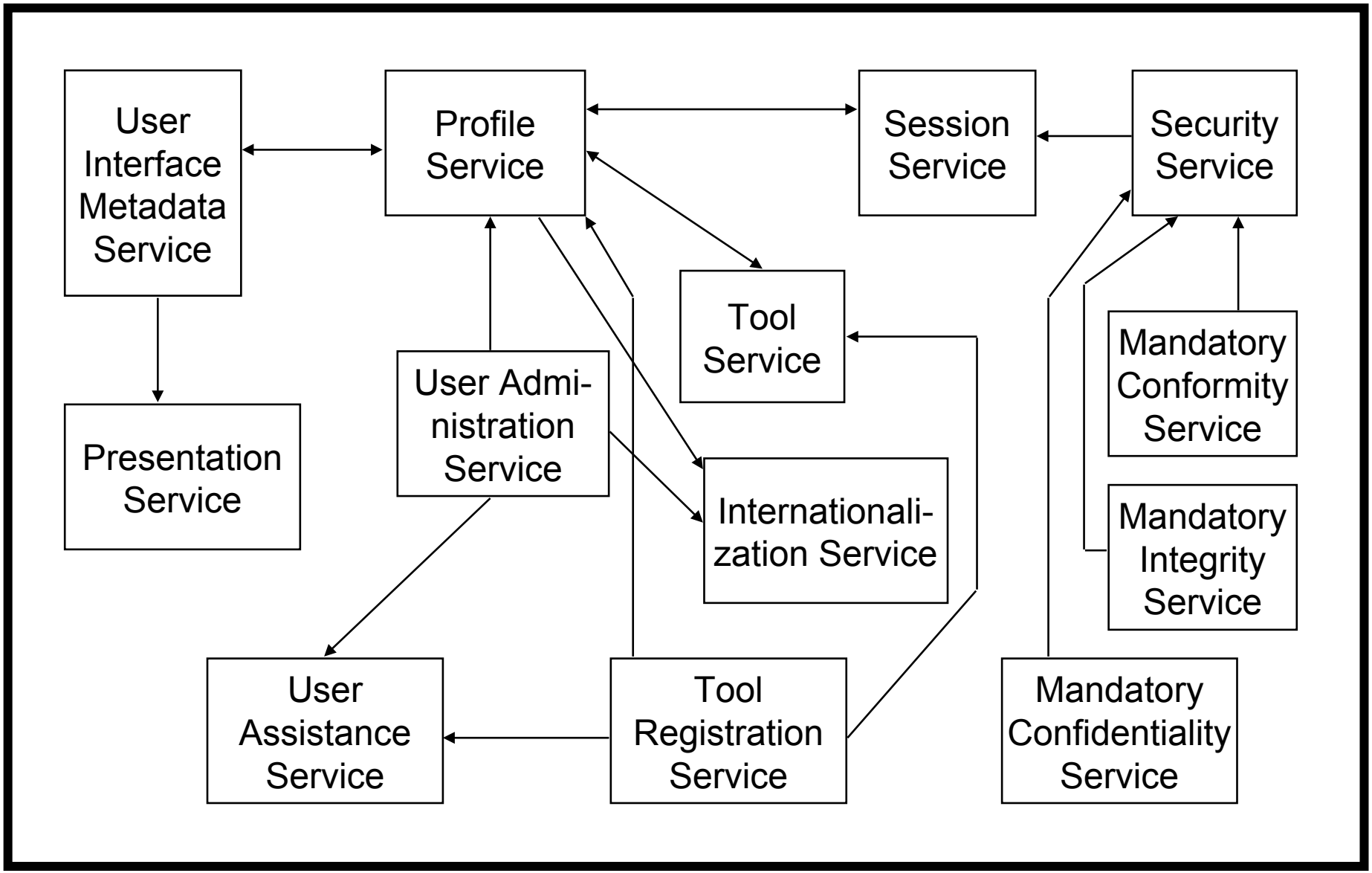
Die **Object Management Services** dienen der Definition, der Speicherung, der Handhabung, der Verwaltung und dem Zugriff auf Objekte/Dokumente (Dateien, Programme, Bibliotheken, Projekte, Geräte usw.):

- **Metadata Service** gestattet die Definition, Steuerung und Handhabung of Schemata und sonstigen Metadaten (Reflektion, Introspektion)
- **Data Storage and Persistence Service** unterstützt das persistente Anlegen und Speichern of Objekten nach der MetadatenbeWriteung
- **Relationship Service** erlaubt die Definition und Handhabung of Beziehungen zwischen Objekten und Objekttypen.
- **Derivation Service** (Bau-Management) legt die Wege fest, welche Objekte of anderen abgeleitet sind (z. B. Generation Objektcode aus Quellcode ähnlich Make-Files).
- **Concurrency Service** sichert den gleichzeitigen Zugriff für Nutzer und Prozesse zum gleichen Objekt der Repository (Transaktionen, Synchronisation)
- **Version Service** unterstützt das Anlegen, Zugreifen und Verbinden of Objekt- und Konfigurationsversionen der SEU.

ECMA: Weitere Services

- Die **Policy Enforcement Services** sind für Sicherheitsaspekte, Integritätsüberwachung und Verwaltungsfunktionen zuständig:
  - **Mandatory Confidentiality Service** legt auf eigenen Wunsch Zugriffsrechte und Sicherheitsanforderungen (geheim, str. geheim) für Objektinfomationen fest.
  - **Mandatory Integrity Service** gestatten den Schutz of SEU-Objekten vor unauthorisierten Änderungen, z. B. Eintragung "read only"usw.
  - **Mandatory Conformity Service** überwacht alle Aktivitäten zur Einhaltung of Konformitätsanforderungen, die z.B. aus der Qualitätssicherung stammen.
- Die **Communication Services** dienen der Kommunikation zwischen Werkzeugen, zwischen Basisdiensten sowie Diensten verschiedener SEU.
  - Basismechanismen sind Nachrichten (Punkt-zu-Punkt, Broadcast, Multicast), Betriebs-systemaufrufe, Remote Procedure Calls und der Datenaustausch
- Die **Framework Administration** und **Configuration Services** übernehmen die sorgfältige Installation der SEU und ihre laufende Pflege u.a.:
  - **Tool Registration Service** übernimmt das An- und Abmelden neuer Tools.
  - **User Administration Service** unterstützt das An- und Abmelden of Nutzern zum System!

ECMA Toaster: Abhängigkeiten zwischen Diensten



30.A.2 Ein Metamodellgesteuertes Framework zur Werkzeugintegration (PCTE)

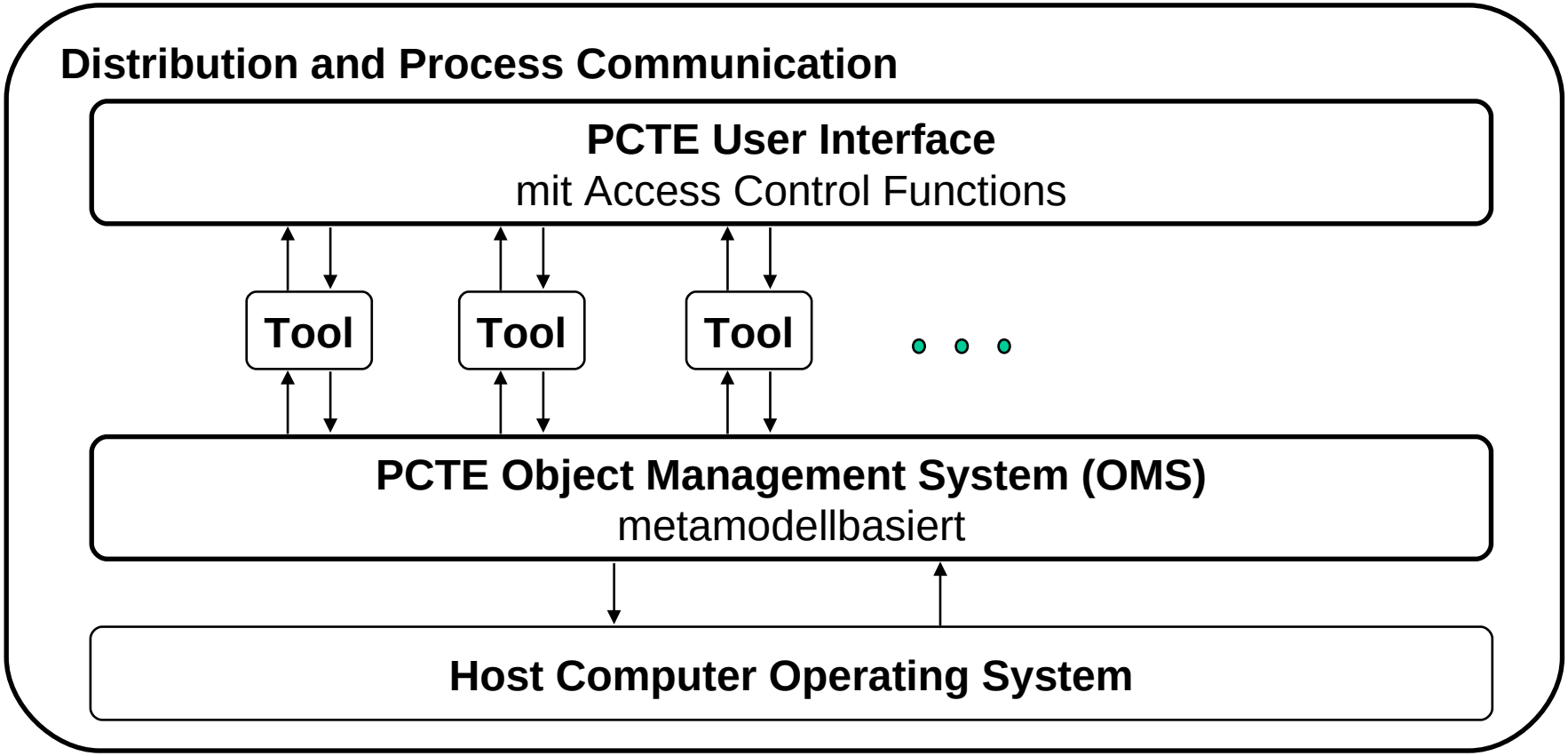
Mit eigenem Technikraum und Metasprache PCTE-OBS

http://ieeexplore.ieee.org/iel3/2107/7595/00313508.pdf?arnumber=313508

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.8315&rep=rep1&type=pdf

Portable Common Tool Environment (PCTE+, HPCTE)

- PCTE ist eines der historisch ersten metamodellgesteuerten Werkzeugintegrationsframeworks
- PCTE erfüllt den Schnittstellenstandard der ECMA - unterstützt systemunabhängigen Zugriff auf Werkzeuge und Repository



Quelle: ECMA - Portable Common Tool Environment (PCTE), Abstract Specification; ECMA-149, 2nd Edition, Juni 1993

Technische Merkmale of PCTE

PCTE stellt eine Menge hochintegrierter Basisdienste bereit, die eine vielseitige Grundlage für verteilte Software-Entwicklungsumgebungen (SEU) bilden

- **verteiltes DBMS** basierend auf dem ERD mit Erweiterungen, wie zusammengesetzte Objekte, Versionen, Mehrfachvererbung, dynamisch kreierte Sichten, eingebettete Transaktionen usw.;
- ein **exklusives Ausführungssystem**, welches Prozeßhierarchien, Vererbung of offenen Files, Prozeßkommunikation über Pipes und Nachrichtenwarteschlangen gestattet. Werkzeuge können als Shell-Skript geschrieben und in mehreren Fenstern unterstützt werden.
- **verteilte Dienste**, d.h. Objektbasis und Prozesse sind transparent verteilt, Replikation of Objekten sowie Schema-Management sind ebenfalls dezentralisiert.
- **erweiterbare Sicherheitsmerkmale**, wie Vertraulichkeit, geschützte Zugriffssteuerung für individuelle Objekte, Revisionsfähigkeit.

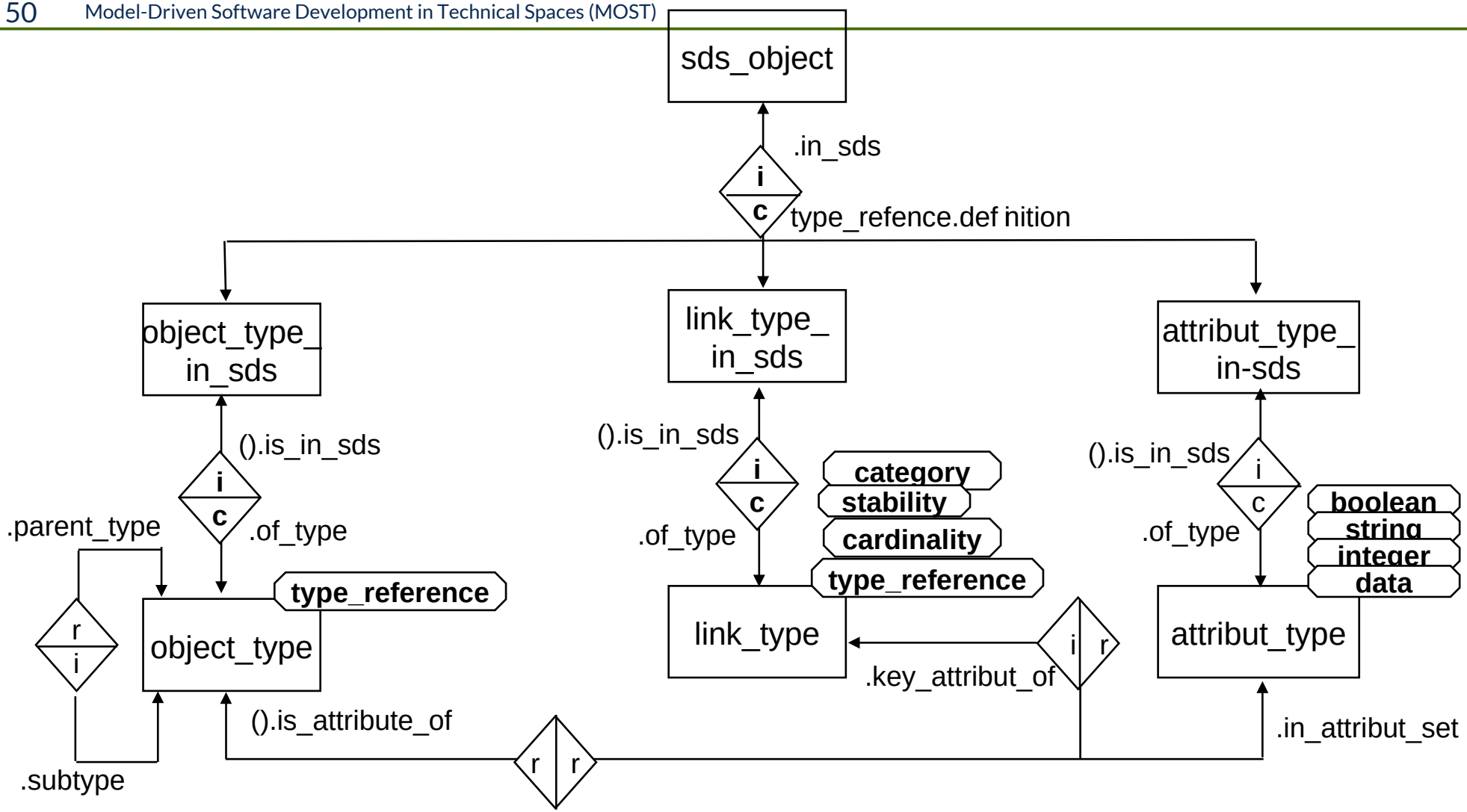
Quelle: http://pi.informatik.uni-siegen.de/pi/hpcte/hpcte.html

PCTE-OMS-Modell (Object Management System)

- Das OMS stellt Datentyp- und Datenspeichermöglichkeiten sowie Concurrency-Control-Mechanismen zur Verfügung,
  - def niert statische Informationen, die in der ***object base*** (Repositorium aller persi-stenten Daten) gehalten werden,
  - liefert Konzepte für die Softwareentwicklung, wie beispielsweise die (Typ-)Verer-bung der objektorientierten Modelle.
- Das OMS ist metamodell-gesteuert. Es enthält **Typdef nitionen**, die in **Schema Def nition Sets (SDS, Metamodellen)** beschrieben werden:
  - **Objekte:** Entitäten, auf denen Operationen der Werkzeuge ausgeführt werden. Instanzen können Dokumente, Textf les, Quell- oder Objektcode, Task aber auch Geräte und Nutzer sein.
  - **Links (Assoziationen):** gerichtete Beziehungen zwischen (Ursprungs-) und (Ziel-)Objekt (bidirektional)
  - **Attribute:** beWriteen Objekte und Links näher. Sie enthalten einen bestimmten Wertetyp und können sowohl Schlüssel- als auch normales Attribut sein.
- Die DDL **PCTE-OMS** ist **geliftet** (selbstreferenzierend**,** in sich selbst spezif ziert, DDL ist geliftet als Metasprache)

Quelle: http://gille.loria.fr:7000/Emeraude/emeraude.html

PCTE-OMS Metasprache ist eine Erweiterung of ERD (Vererbung, Komposition)

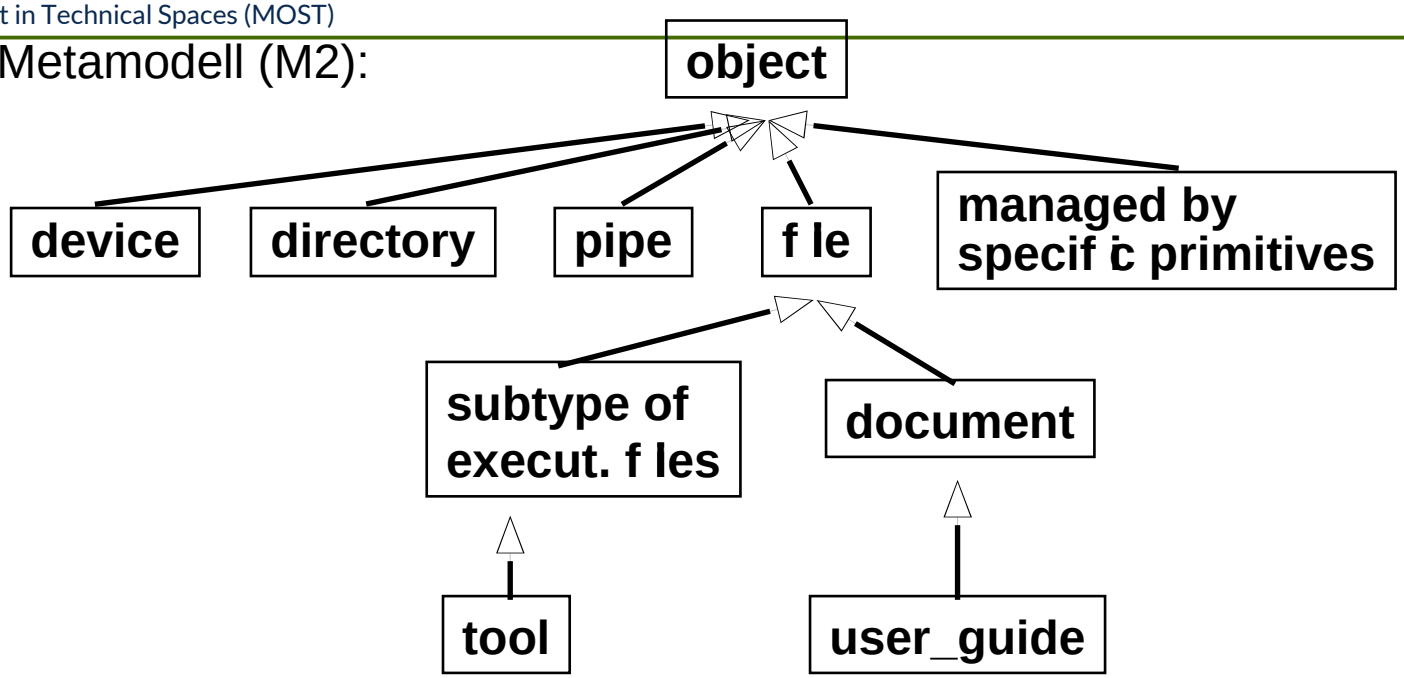


Link-Typ Kategorien: ***c composition***
r reference
i implicit
s system implicit,

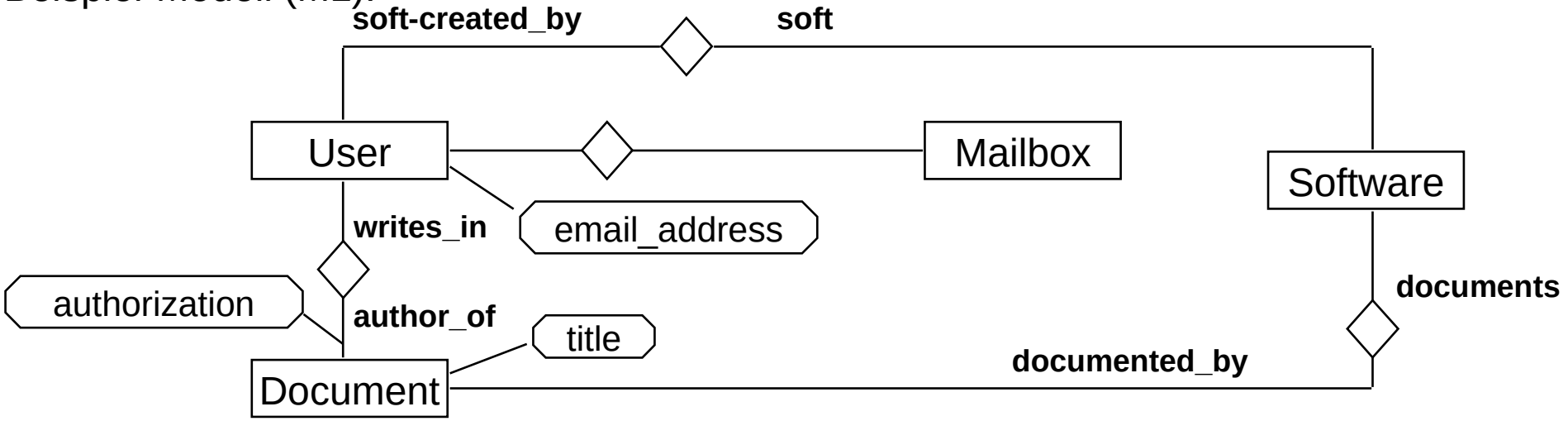
bei Anlegen eines neuen Objekts in Referenz zum existierenden.
zwischen Ursprungs- und Zielobjekt
kann z.B. reverse link für einen angelegten Link sein
automatisch vom System (PCTE-OMS) gesetzt.

PCTE-Objekt-Strukturen mit erweitertem ERD

PCTE DDL Metamodell (M2):

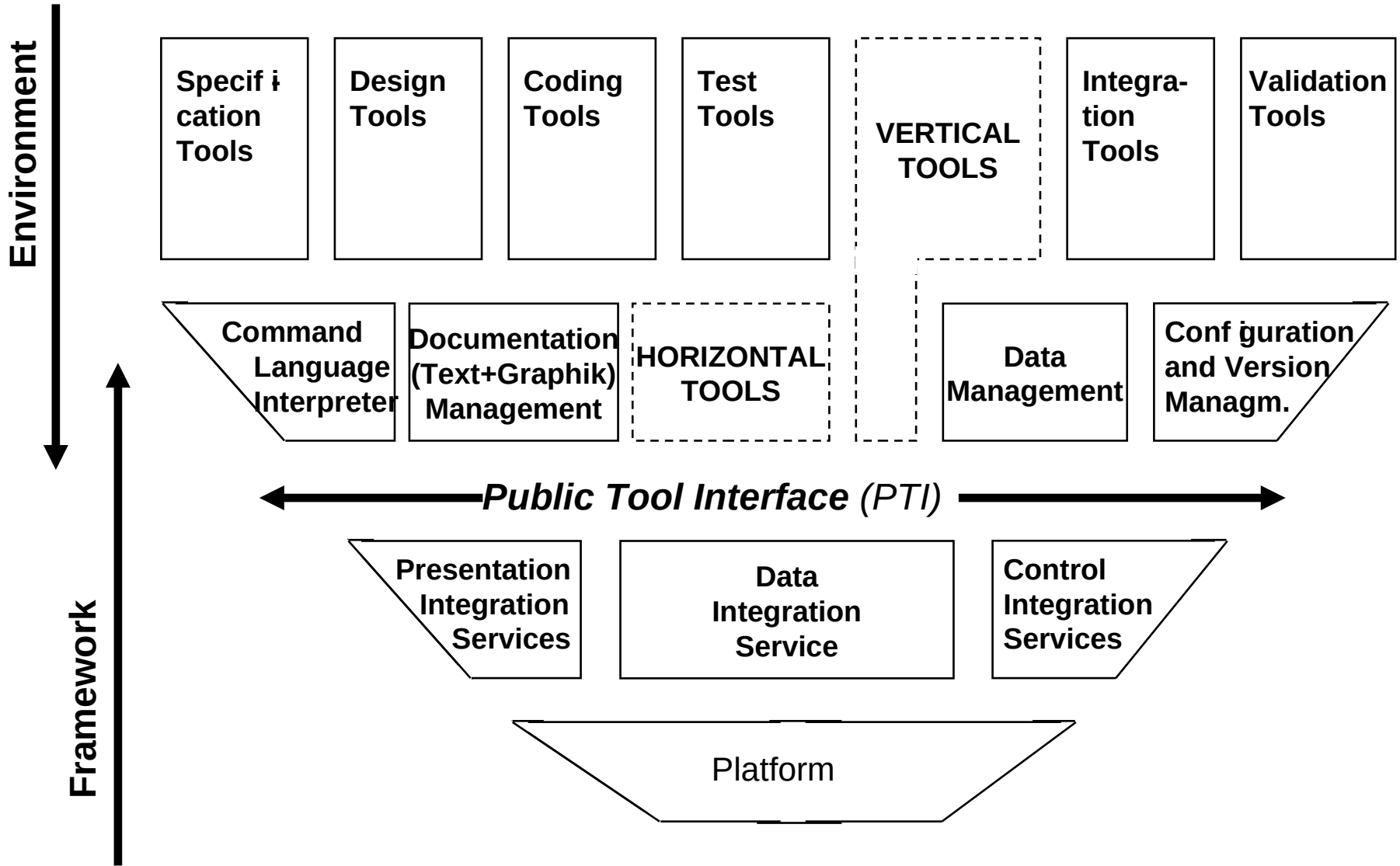


Beispiel-Modell (M1):



Emeraude PCTE Framework

http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=182066
http://www.springerlink.com/content/g111t41326211512/



More PCTE Implementations

- PACT PCTE Implementation
  - Thomas, Ian. Tool integration in the pact environment. In Proceedings of the 11th International Conference on Software Engineering, pages 13-22, May 1989.
- HPCTE implementation of University of Siegen
  - Java API
  - Supports views on the repository
  - http://pi.informatik.uni-siegen.de/pi/hpcte/hpcteapps.html