

III. Integration und Komposition von Werkzeugen

30. Integration von Werkzeugen, Austauschformate und Software-Entwicklungsumgebungen

1

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und
Multimediatechnik
<http://st.inf.tu-dresden.de>
Version 13-0.2, 02.12.13

- 1) Werkzeugintegration
- 2) Datenintegration
- 3) Architektur von SEU
- 4) ECMA-Referenzmodell
- 5) Austauschformate und Technikraum-
Brücken
- 6) Frameworks zur Werkzeug-
integration (PCTE)

Referenzen

2

- ▶ 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>

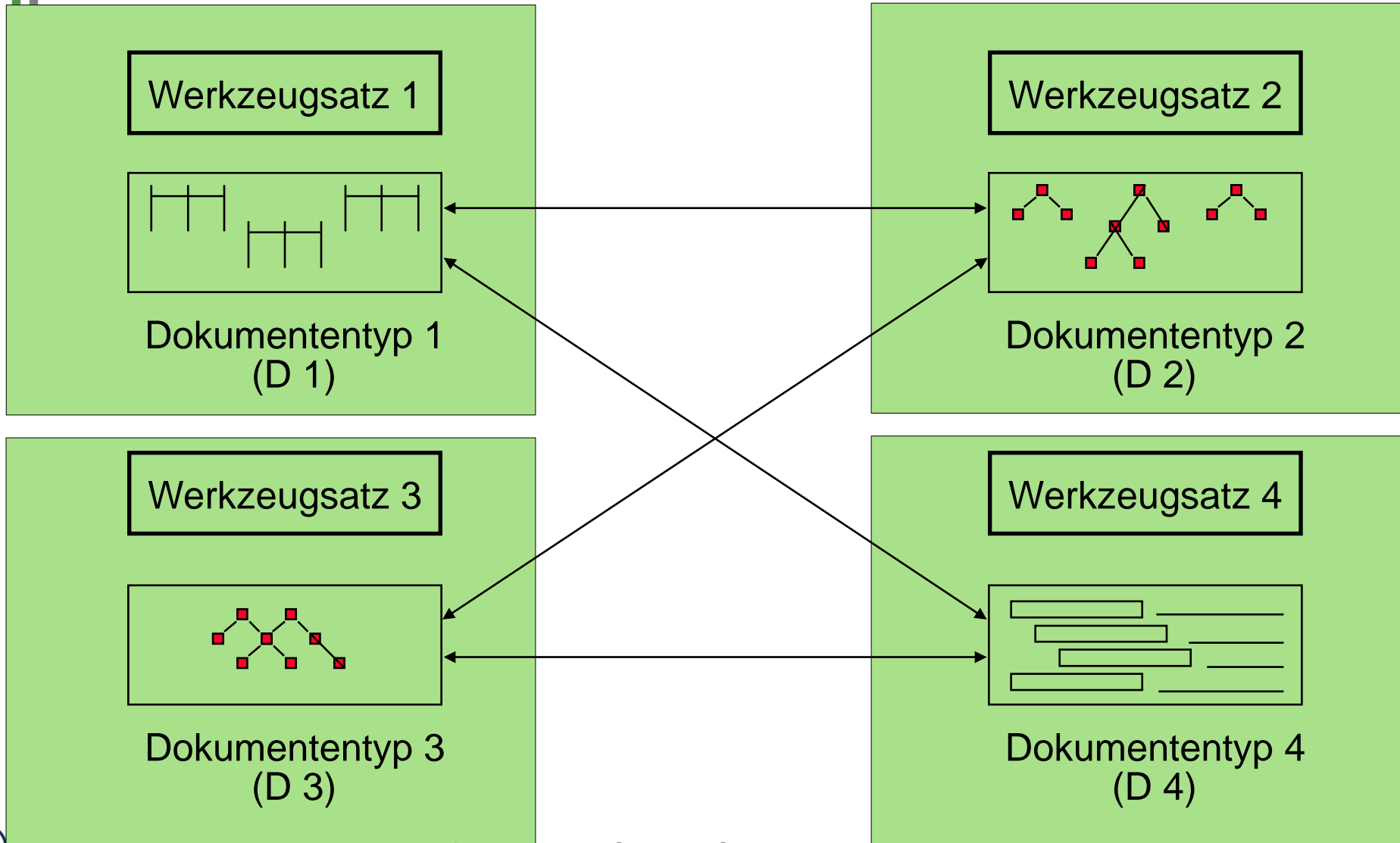
30.1 Konzepte der Werkzeug- integration



3

Integration von Werkzeugsätzen zur Softwareentwicklung

4



Werkzeugintegration

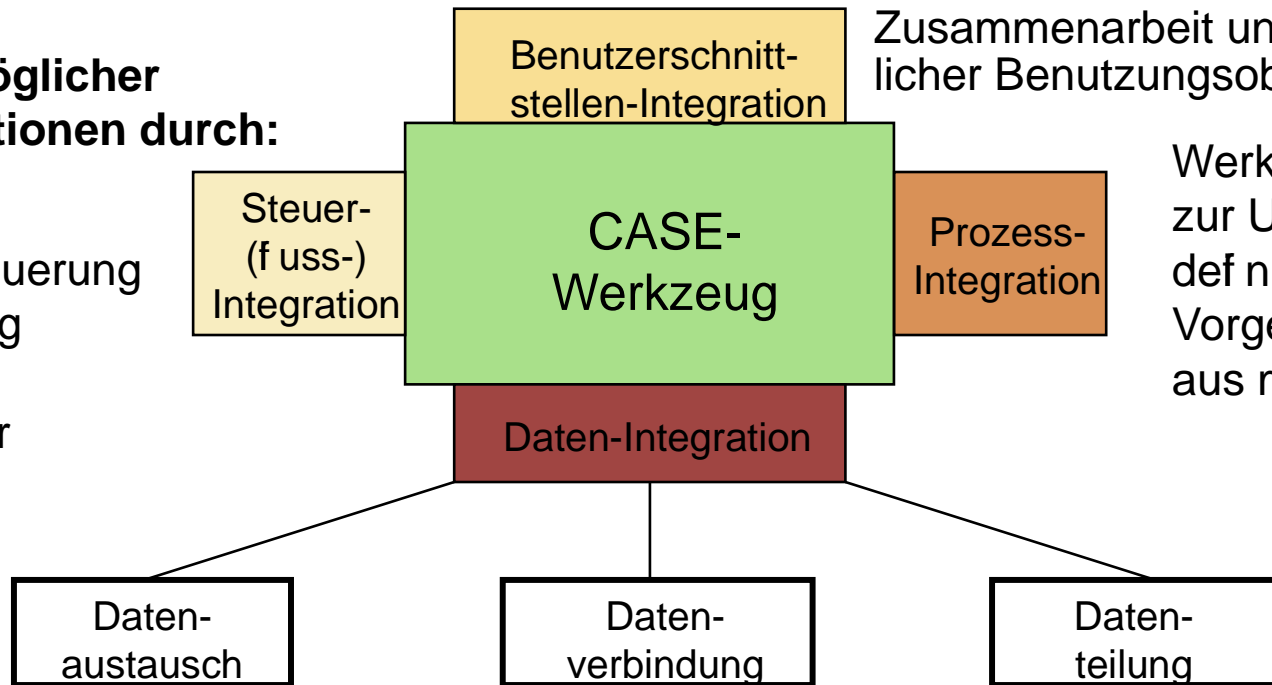
5

Zusammenarbeit mehrerer Werkzeuge mit gleicher Präsentation, gleichem Verhalten und Interaktionsformen

Verbindung möglicher Werkzeugfunktionen durch:

Prof. U. Armann, Softwareentwicklungswerkzeuge (SEW)

- Interprozesssteuerung
- Metasteuerung (Aufruf des Tools über „Automaten“)



Werkzeugintegration zur Unterstützung eines definierten Prozesses/ Vorgehens bestehend aus mehreren Schritten

- Forderungen:**
- Interoperabilität mit Formatwandlung
 - Redundanzfreiheit - Einmalspeicherung
 - Datenkonsistenz und Persistenz
 - Synchronisation - Abstimmung des Werkzeugzugriffs

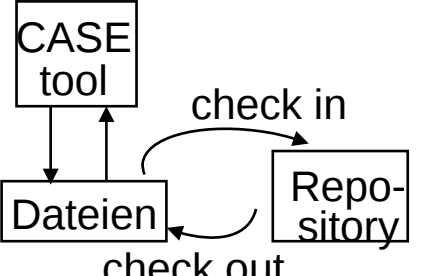
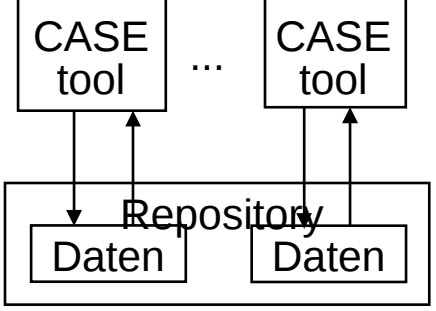
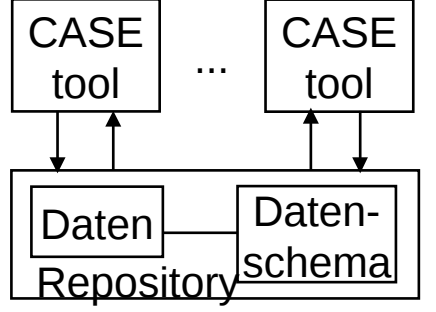
30.2 Datenintegration



6

Repository - Datenintegrationsstufen

7

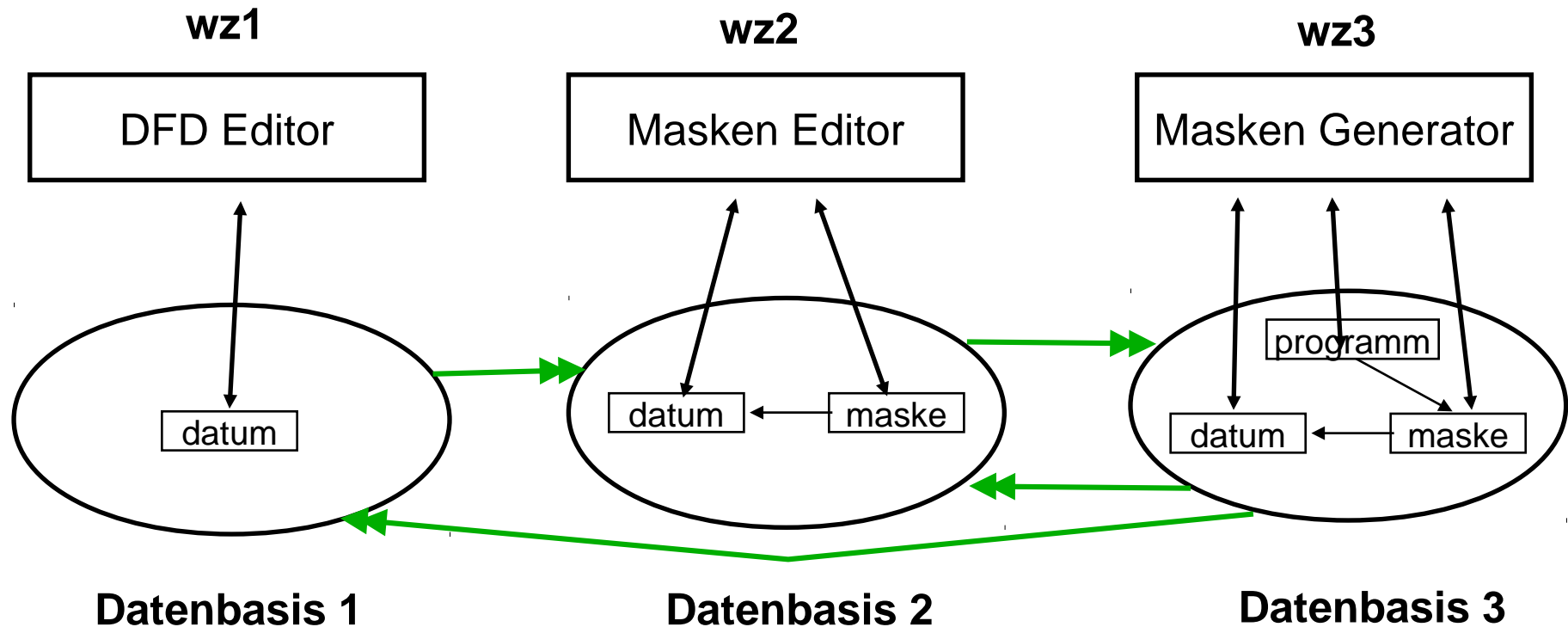
Integrationsart	Schema	Wirkungsweise
<p>Black-box-Integration (schwach)</p>		<ul style="list-style-type: none"> • Werkzeug arbeitet isoliert auf eigenen Datenstrukturen • Repositorium stellt Daten bereit (check out) • Nach Arbeit Ablage in Repositorium (check in) • checkout/in oft manuell
<p>Grey-box-Integration (mittel)</p>		<ul style="list-style-type: none"> • Werkzeugzugriffe durch Repository-Dienste ersetzt • Unterstützung von Datenintegrität und Interoperabilität zwischen Werkzeugen • Keine Offenlegung essentieller Bestandteile der Werkzeug-Implementierung • Verkapselung in einem Zugriffsmodul
<p>White-box-Integration Datenteilung (stark)</p>		<ul style="list-style-type: none"> • Definition einheitl. Datenschema für alle Werkzeuge • Alle Werkzeuge setzen über Zugriffsdienste auf • Einfache Sicherung von Konsistenz, Datenintegrität und Datensicherheit • Werkzeuge sind bei Änderung an Datenschema anzupassen



Werkzeuge mit Datenaustausch (ad-hoc), ohne Datenverbindung und -integration

8

- ▶ Keine gemeinsamen Daten, hoher (manueller) Aufwand zum Austausch
- ▶ Austausch mit Datenfluss-Strömen (Datenflussarchitektur)
 - Querysprachen filtern den Datenaustausch
 - Datenformate werden in einer Austauschsprache definiert
- ▶ Aber: unabhängiges, paralleles Arbeiten möglich

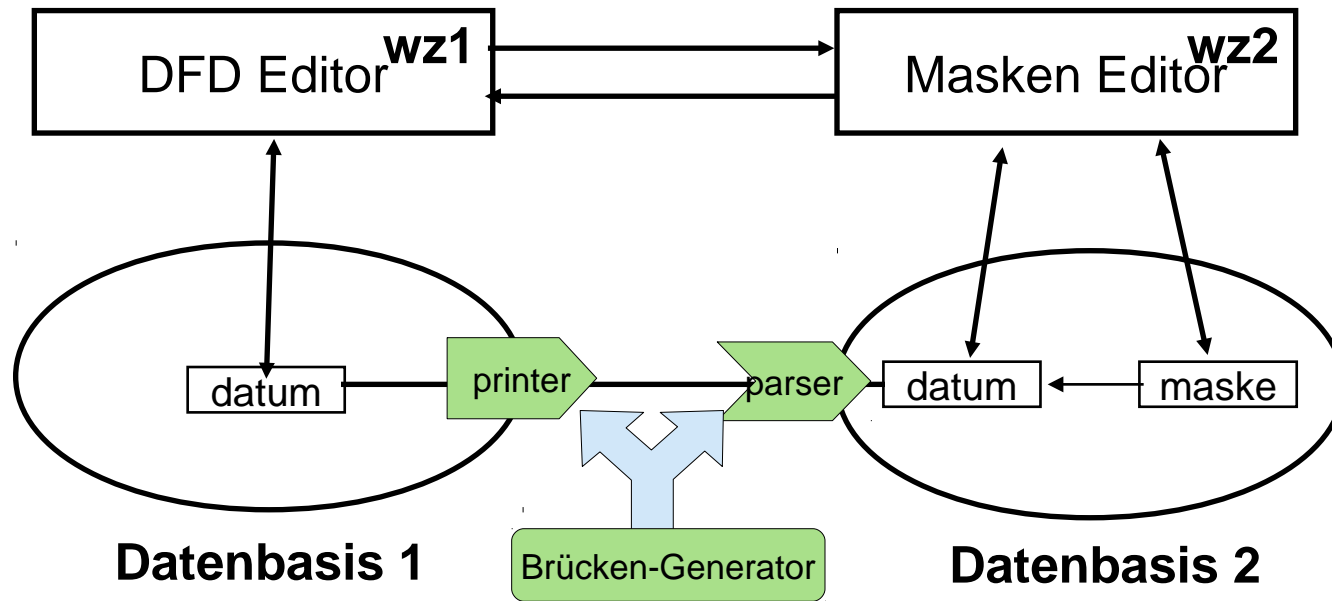


Quelle: nach [HMF. S. 196]

Datenverbindung durch Datenaustausch (Transformationsbrücken)

9

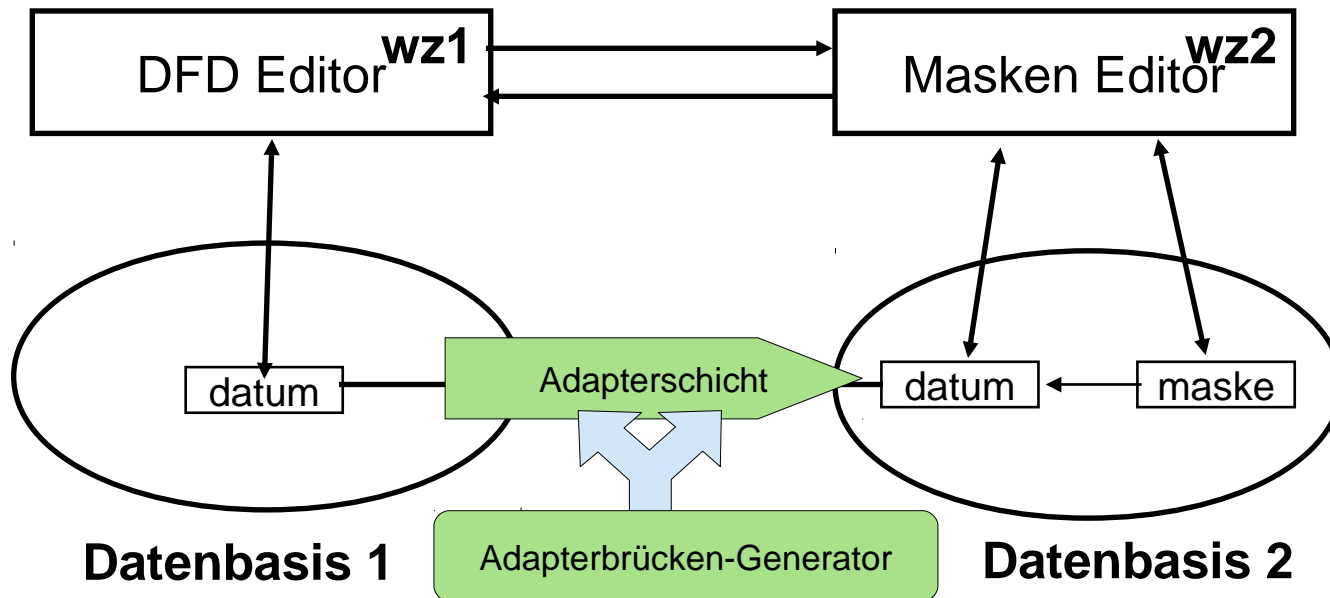
- ▶ **Datenverbindung** ist die Einführung semantischer Beziehungen zwischen Teilen von Datenbeständen
- ▶ **Automatisierter Datenaustausch** ist die automatisierte Übertragung von *semantisch verbundenen Daten* zwischen Werkzeugen in standardisierten **Austauschformaten** (z. B. ASN, XMI, CDIF, XML)
 - Automatisierung beruht auf Metamodellen
- ▶ **Transformationsbrücke:** Prettyprinter transformieren das interne Format eines Repositoriums in ein externes
 - Parser wandeln es in das interne Format des anderen Werkzeugs
 - Querysprachen filtern den Datenaustausch



Datenverbindung durch Datenaustausch (Adapterbrücken)

10

- ▶ Eine **Adapterbrücke** bildet eine Schicht zwischen zwei Datenablagen, die den inkrementellen Zugriff von einem auf das andere erlaubt
 - Auch hier müssen die Daten gewandelt werden, aber das geschieht inkrementell und meist ohne Austauschformat



Aktuelle Austauschformate

11

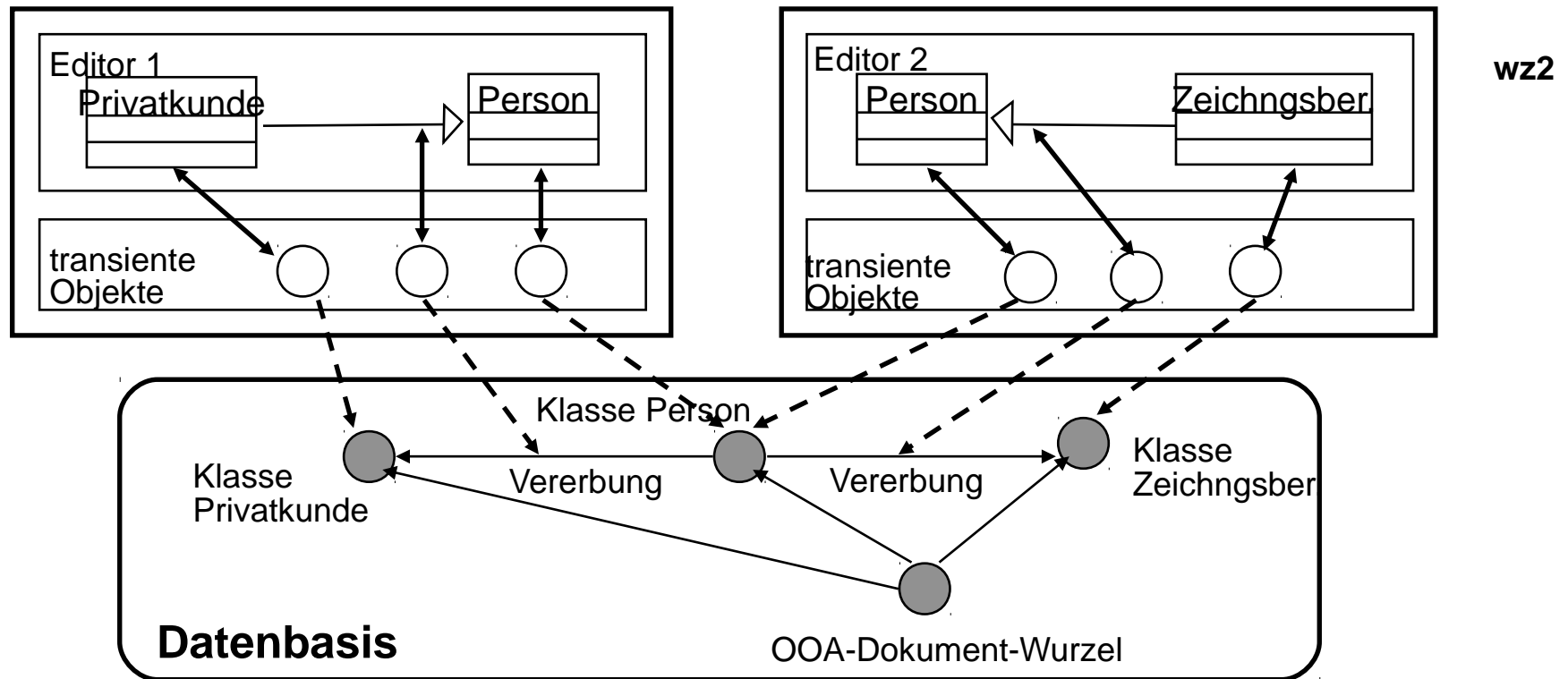
Austauschformat: Hersteller- und methodenunabhängiges Datenkonzept für die Modellierung von Austauschdaten zwischen Werkzeugen

- ▶ Comma-separated values (CSV): einfaches text-basiertes Austauschformat für Werkzeuge auf Relationen und Tabellen (Excel, TeX, ...)
 - Keine Metasprache, einfaches Tabellenschema <http://tools.ietf.org/html/rfc4180>
 - http://en.wikipedia.org/wiki/Comma-separated_values
- ▶ CASE Tool Data Interchange Format (CDIF) - Metasprache ERD für Data Definition, aber auch
 - 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) zum Austausch von UML-Diagrammen im XML-Format
 - Meta Object Facility (MOF) als Metasprache
 - <http://www.omg.com/technology/documents/formal/xmi.htm>
- ▶ ASN.1 Standard
 - Eigene Metasprache, an BNF angelehnt
 - http://de.wikipedia.org/wiki/Abstract_Syntax_Notation_One
- ▶ RDF/RDFS Resource Description Format – Modelle als Graphen, gespeichert in elementaren Tripeln <http://www.w3c.org>
- ▶ GXL Graph Austauschformat: Open Source Format zum Austausch von Graphen <http://www.gupro.de/GXL/>

Datenteilung (Gemeinsames Repository)

12

- ▶ Bei **Datenteilung** greifen Werkzeuge direkt auf gemeinsame Daten zu, die in einem gemeinsam verfügbaren Datenbasis (Repository) mit einheitlichen logischen Daten-Schema abgelegt sind.



30.3 Datenverbindung mit Austauschformaten und Technikraum-Brücken

13

Einsatz in Transformations-Brücken zwischen Repositorien

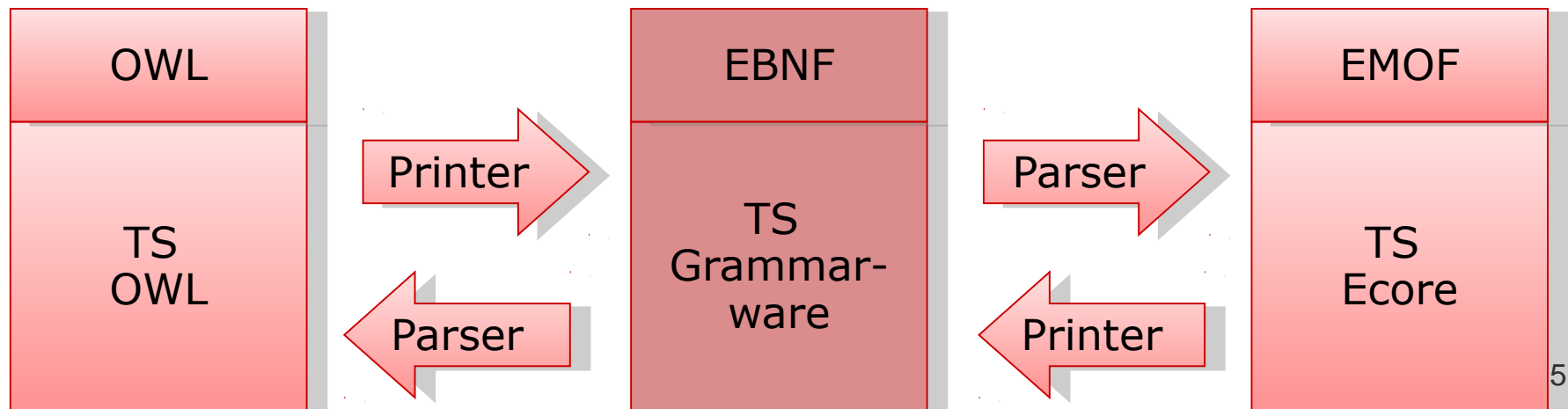


- ▶ **Datenverbindung** zwischen Repositorien beruht auf einer *semantischen Beziehung der Daten*
 - Z.B. Gleiche Sprache auf M2 ermöglicht Mapping zwischen Modellen auf M1
 - Sprach-Abbildung auf M2
- ▶ Für Datenverbindung ist als Austauschformat von jeher konkrete textuelle Syntax benutzt worden (mittels Technikraum Grammarware, mit Metasprache EBNF)
 - Parser lesen den Text und wandeln ihn in das interne Format
 - Prettyprinter schreiben das interne Format um auf die konkrete Syntax
- ▶ Man spricht von **normativer konkreter Syntax**, wenn diese normiert ist, also nicht beliebig

Transformative TS-Brücken mit konkreter Syntax

15

- ▶ Eine **transformative Technikraum-Brücke** (TS-Brücke, TS bridge) bietet
 - ein Austauschformat in konkreter Syntax (via dem TS Grammarware)
 - eine Generierungstechnik für Printer und Parser
- ▶ Am besten: normative konkrete Syntax
 - EMFText: normative konkrete Syntax for Ecore
 - Xtext: normative konkrete Syntax for Ecore and OAW
 - CDIF: normative konkrete Syntax für ERD



Austauschformat CDIF: CASE Data Interchange Format

16

- ▶ CDIF ist ein Austauschformat für CASE-Werkzeuge, basierend auf
 - Metasprache auf M3: ERD
- ▶ Austauschformat einfachen, transparenten Aufbaus zwischen (Modell-)Tools und Repositories
 - hersteller- und methodenunabhängig
 - unterstützt die Kooperation zwischen verschiedenen Tools und Projekten

Textuelle CDIF-Beschreibung eines DFD

17

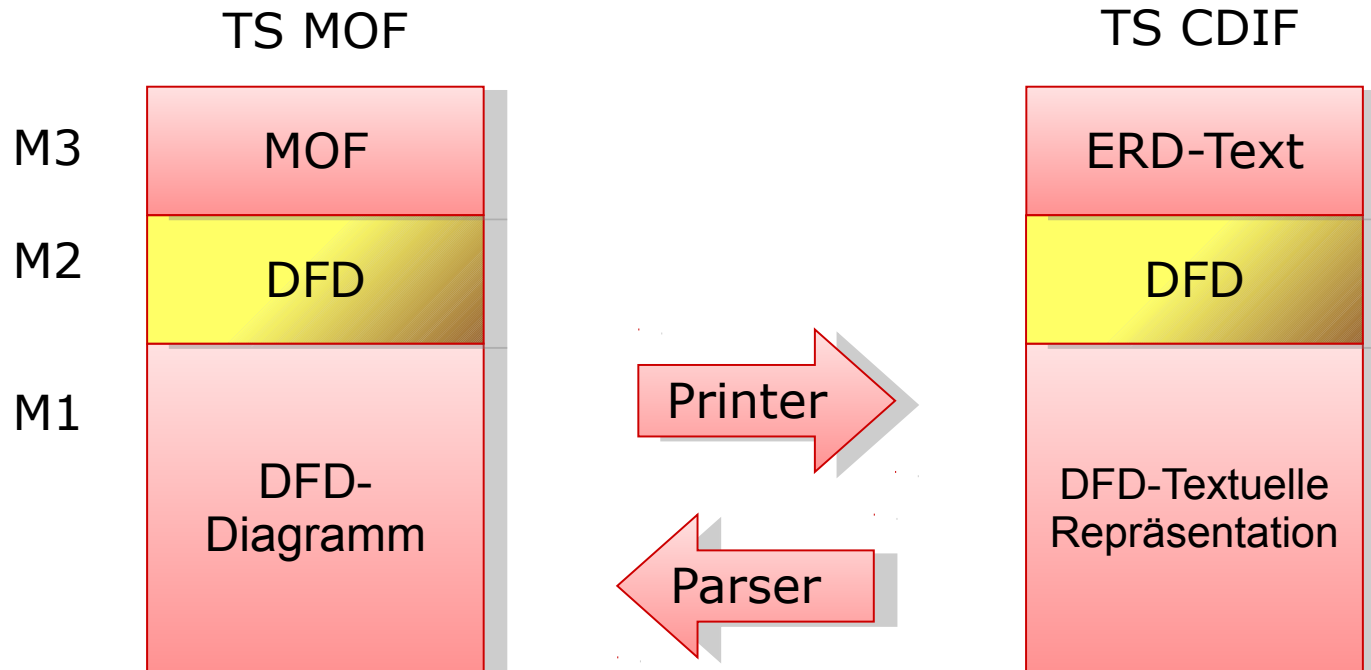
- ▶ Der Standard CDIF nutzt zur Spezifikation von Grammatiken nicht EBNF, sondern eine textuelle Notation von ERD (ERD-Text).
- ▶ Beispiel: DFD-Spezifikation in ERD-Text (Schlüsselwörter in boldface, definierte Nichtterminale in typewriter, benutzte Nichtterminale in italics):

```
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 )
```

Austauschsyntax CDIF

18

- ▶ CDIF definiert eine textuelle Syntax für ERD (ERD-Text)
 - normative textuelle Repräsentation (für alle Sprachen auf M2 gleich)

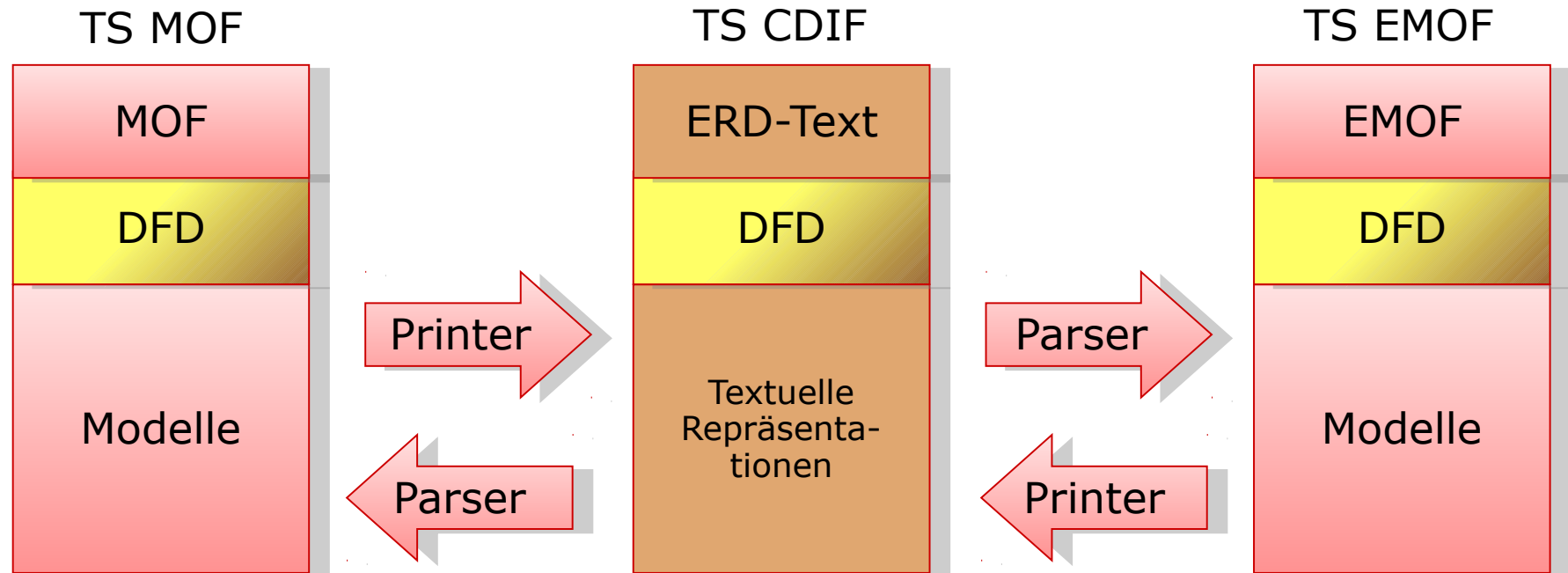


Transformative TS-Brücken via CDIF

19

- ▶ TS-Brücken (via CDIF) generieren Parser und Printer
- ▶ Normative konkrete Syntax in der CDIF-Spezifikation festgelegt

Prof. U. Arßmann, Softwareentwicklungswerkzeuge (SEW)



Austauschformat XML

XML Metadata Interchange Format

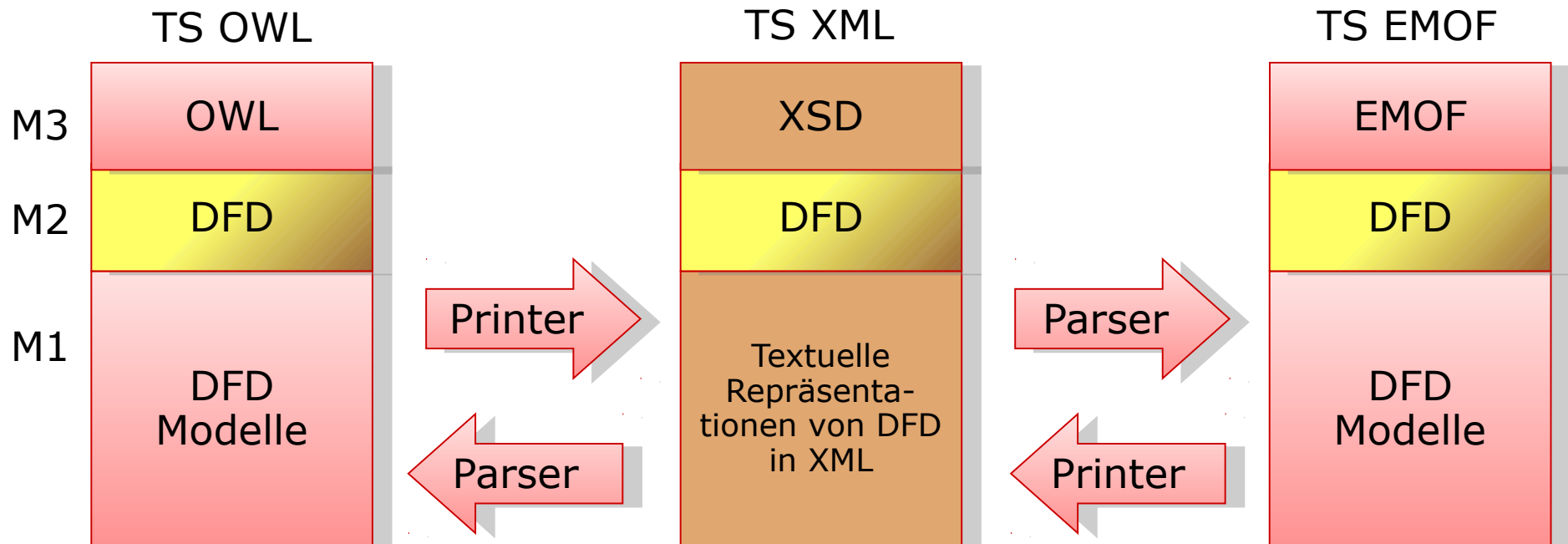
20

- ▶ Ziel: anbieterneutrales offenes Austauschformat für Metadaten/Modelle in verteilten Umgebungen
 - Metasprache MOF
 - generisches „Stream“-Format
 - lose gekoppelte Architektur, einfach für Anbieter zur Verarbeitung in aktuellen Produkten
 - überwindet Lücken zwischen inkompatiblen Tools, Repositories und Anwendungen
- ▶ Stand: OMG-Standard für XML Metadata Interchange (XMI) zwischen Repositories, Tools und Anwendungen Version 2.1 (formal/2005-09-01)
- ▶ Allerdings:
 - Wegen des Indeterminismus des spannenden Baumes keine volle Kompatibilität möglich

Transformative TS-Bridges via XML

21

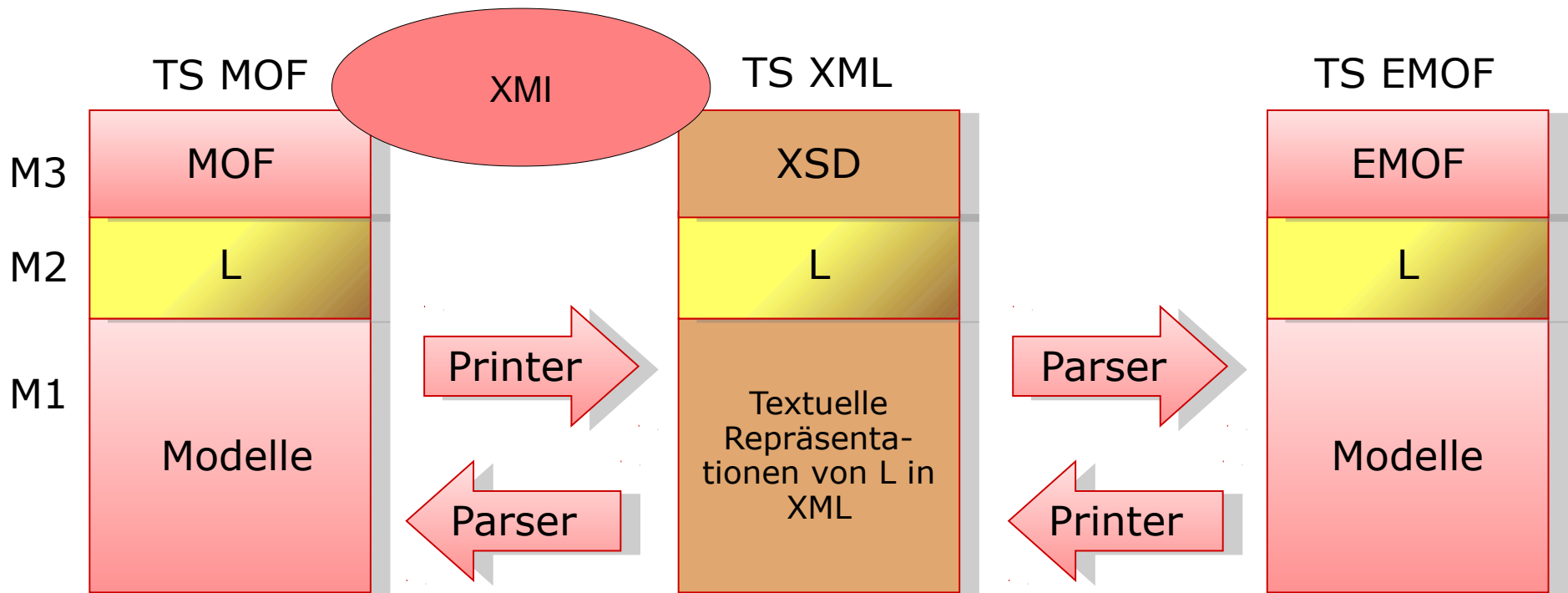
- ▶ XML is a normalized concrete syntax
 - Because of trees, a linearized normalized concrete syntax is possible
- ▶ Good for exchange!



XMI: Transformative TS-Brücke

22

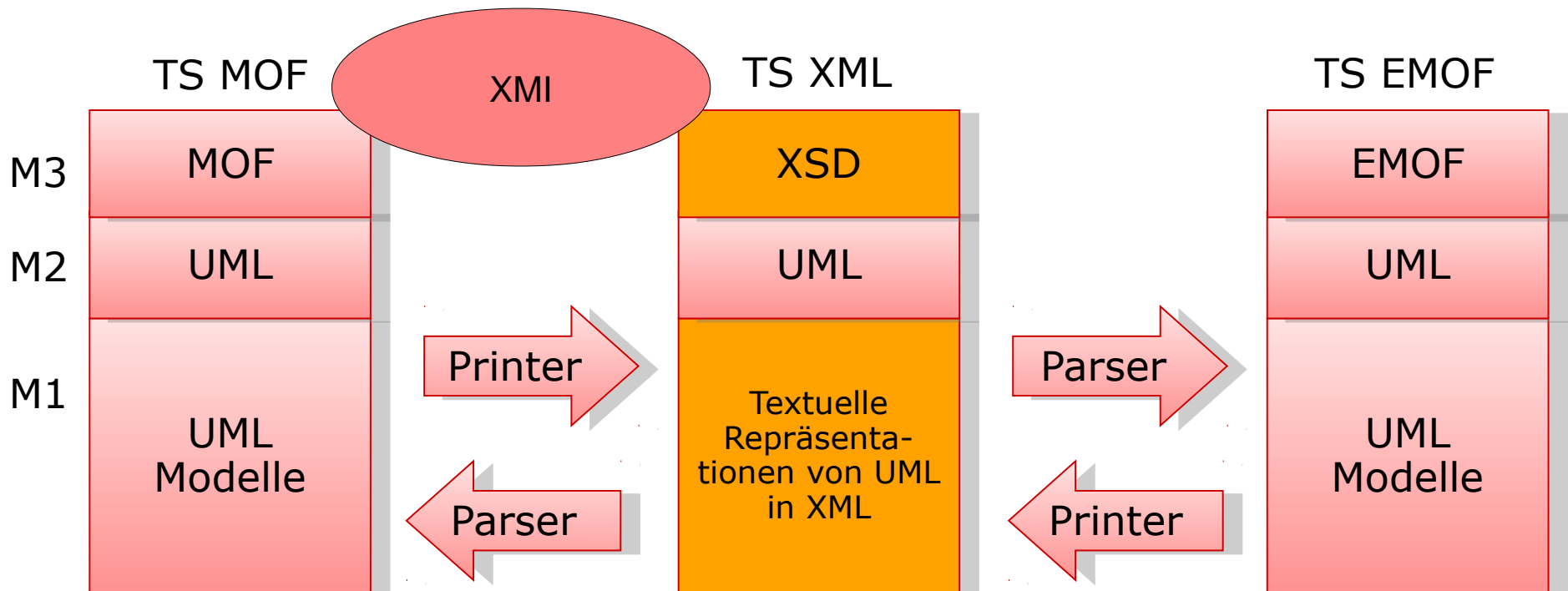
Im allgemeinen Sinne ist XMI eine Brücke zwischen MOF und anderen TS via XSD/XML



XML: Transformative TS-Bücke für UML

23

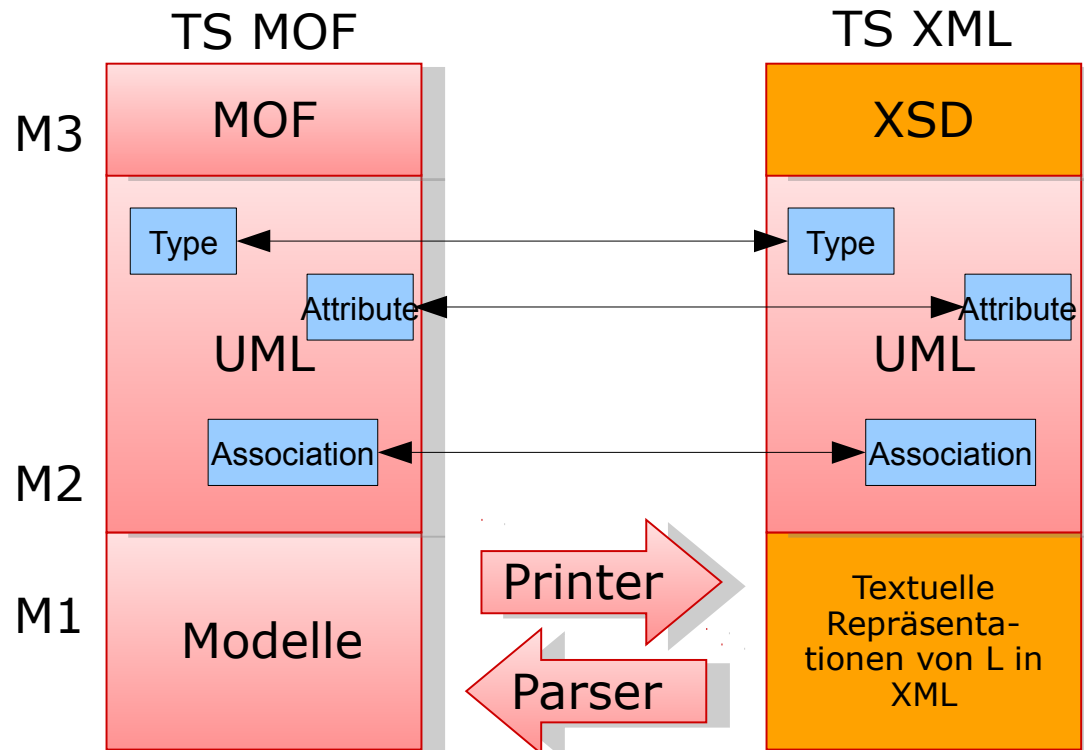
Meist wird allerdings das nur für UML ausgeprägt. Dann ist XMI eine UML-Brücke



Zusammenhang XMI - UML

24

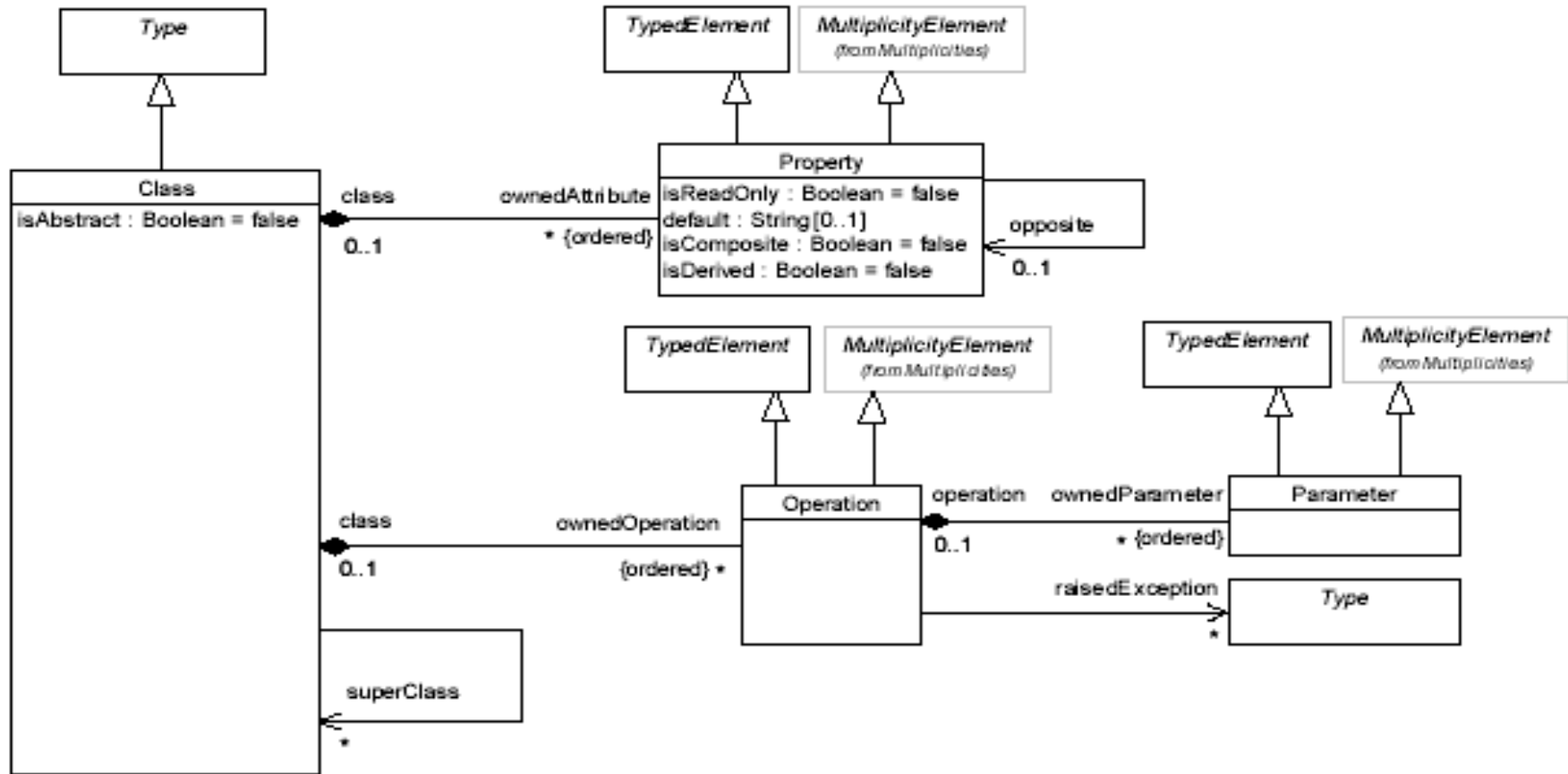
- ▶ XMI liegt ein Metamodell der UML zugrunde, das zweimal, in MOF und XSD, spezifiziert wird
 - Zwischen beiden Metamodellen wird ein **Sprachabbildung (language mapping)** angegeben
 - Aus diesem werden Printer und Parser generiert



Erinnerung: Classes and Properties in UML

25

Definieren einer Klasse als Typ und Festlegung der weiteren Elemente zur klassenbasierten Modellierung



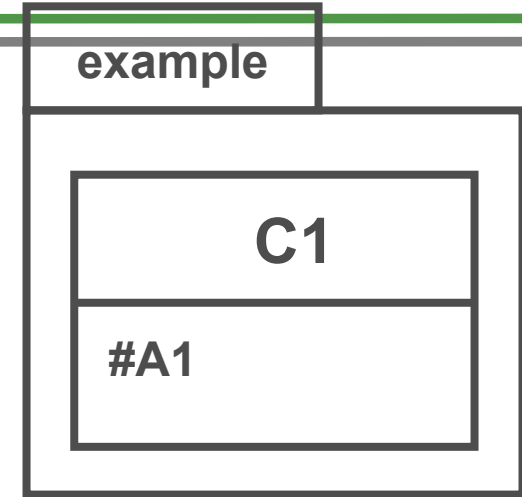
Quelle: UML 2.0 Infrastructure Specification; OMG Adopted Specification ptc/03-09-15



Beispiel einer XMI-Objektinstanz: Kodierung einer UML-Klasse

26

```
<?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>
```



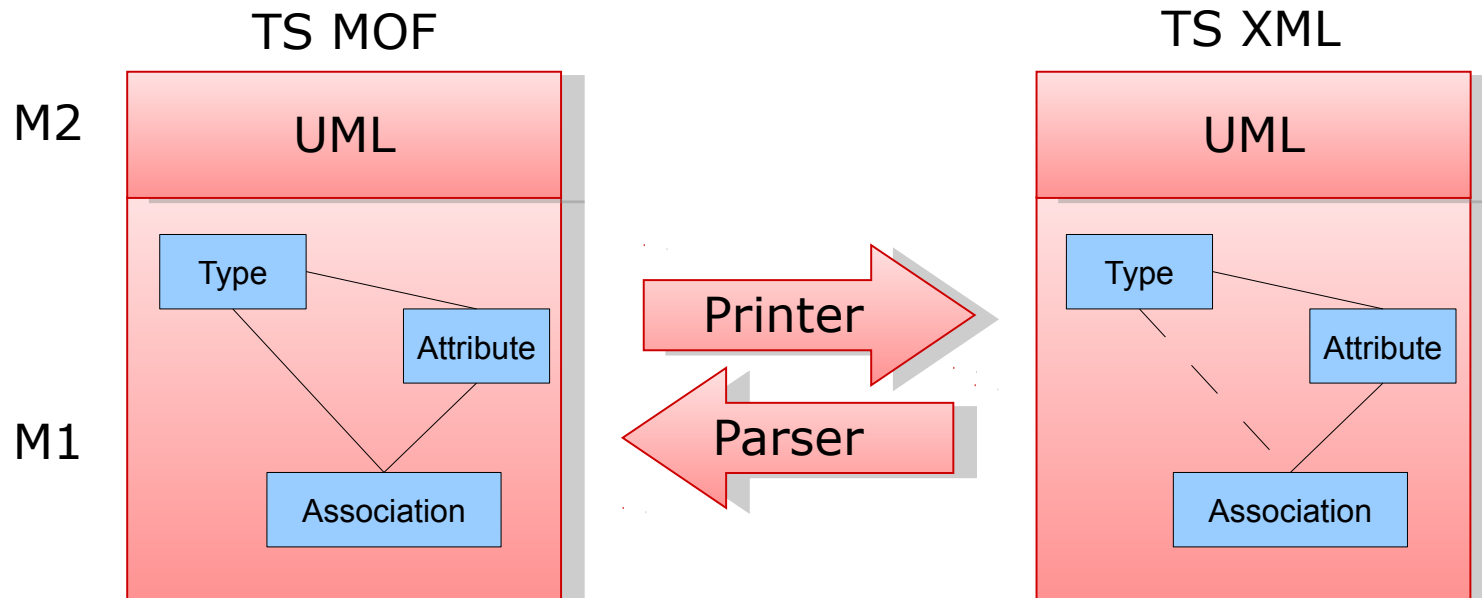
UML Typen

(ähnliches Beispiel siehe : www.jeckle.de/xmi_ex1.htm)

Problem: normativer spannender Baum für Graphmodelle

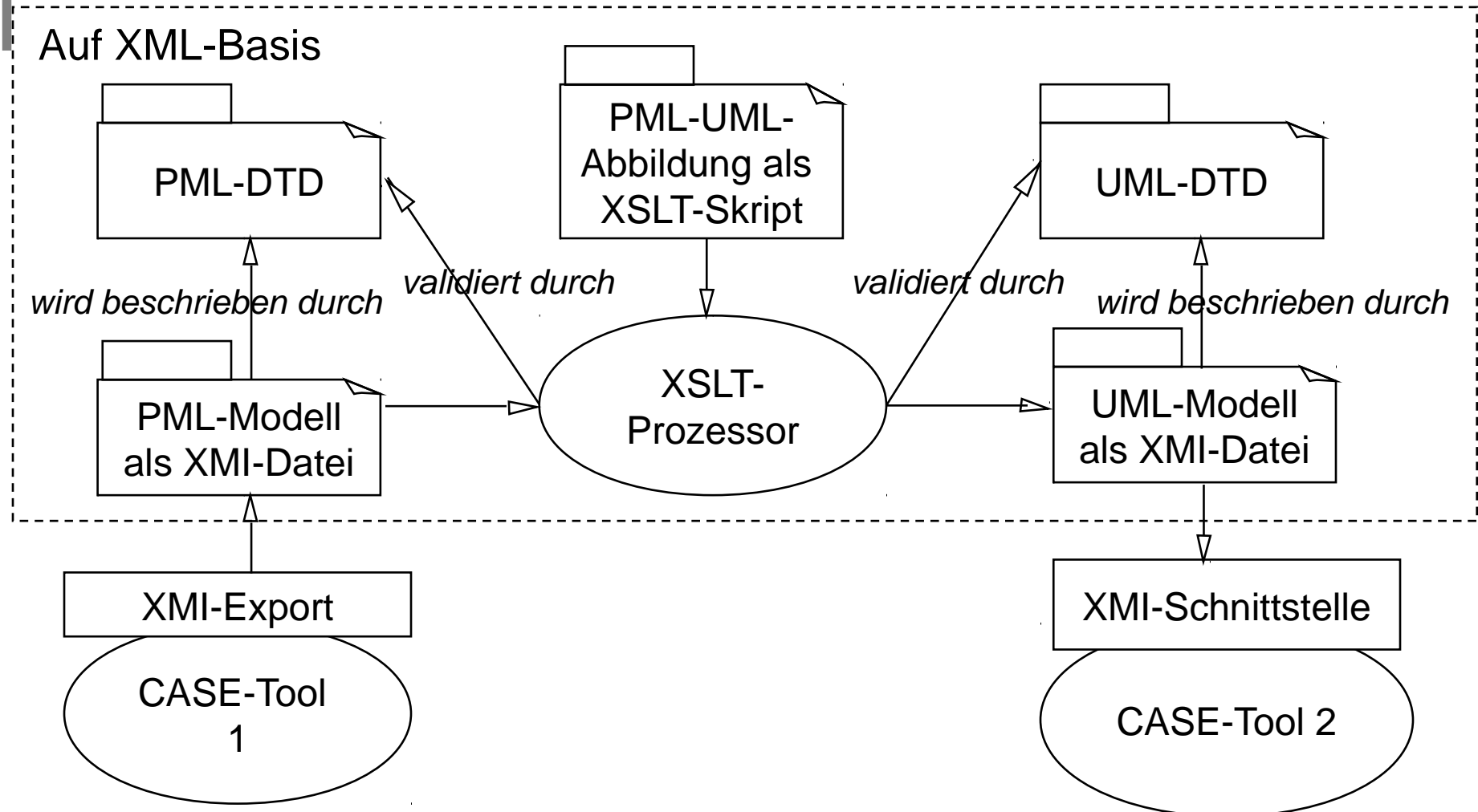
27

- ▶ UML bzw. MOF sind graphbasiert, XML baumbasiert
- ▶ XML muss bestimmte Links in Namensreferenzen auflösen
 - Dazu wird über UML oder MOF-Modell ein Spannender Baum gelegt (z.B. entlang der Aggregation)
 - Alle Links, die nicht im spannenden Baum vorkommen, werden mit Namensreferenzen dargestellt, und **nicht** im XML Baum
- ▶ Da der spannende Baum nicht deterministisch ist, entstehen Inkompatibilitäten



Bsp.: Adaption von XML-basierten Modellen mit XML-DML wie XSLT

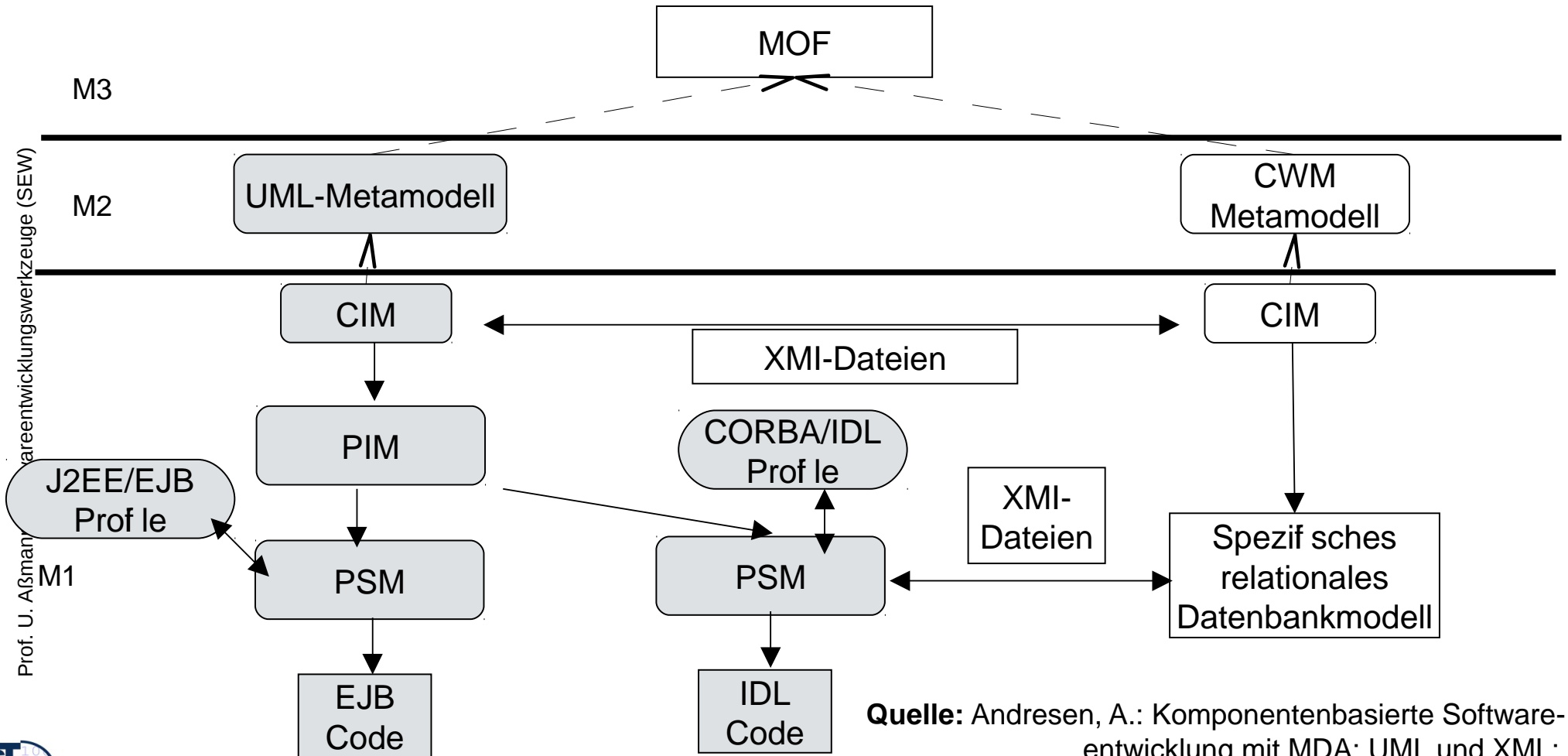
28



Bsp.: Datenaustausch mit XML für CIM im Kontext von Model-Driven Architecture (MDA)

29

Computation independent model (CIM) ist eine Requirements-Spezifikation



Quelle: Andresen, A.: Komponentenbasierte Softwareentwicklung mit MDA; UML und XML; Hanser Verlag 2003, S. 251

Bsp. Transformationsbrücke zwischen ADO und OWL

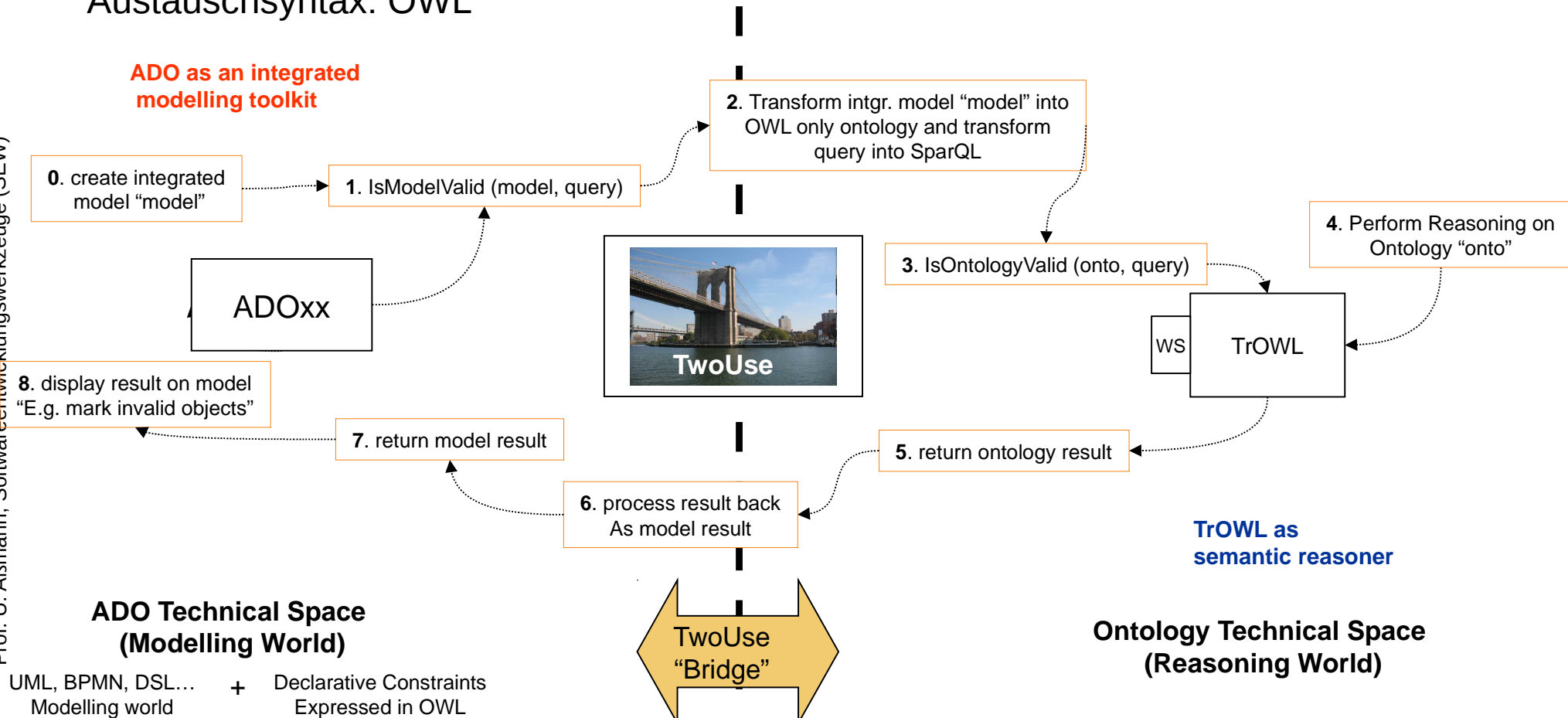
30

TwoUse (U Koblenz) ist eine Transformationsbrücke zwischen TS ADO (BOC Wien) und TrOWL (OWL, Uni Aberdeen)

Austauschsyntax: OWL

ADO as an integrated modelling toolkit

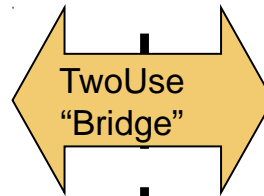
Prof. U. Altmann, Softwareentwicklungswerkzeuge (SEW)



ADO Technical Space (Modelling World)

UML, BPMN, DSL... + Declarative Constraints Expressed in OWL

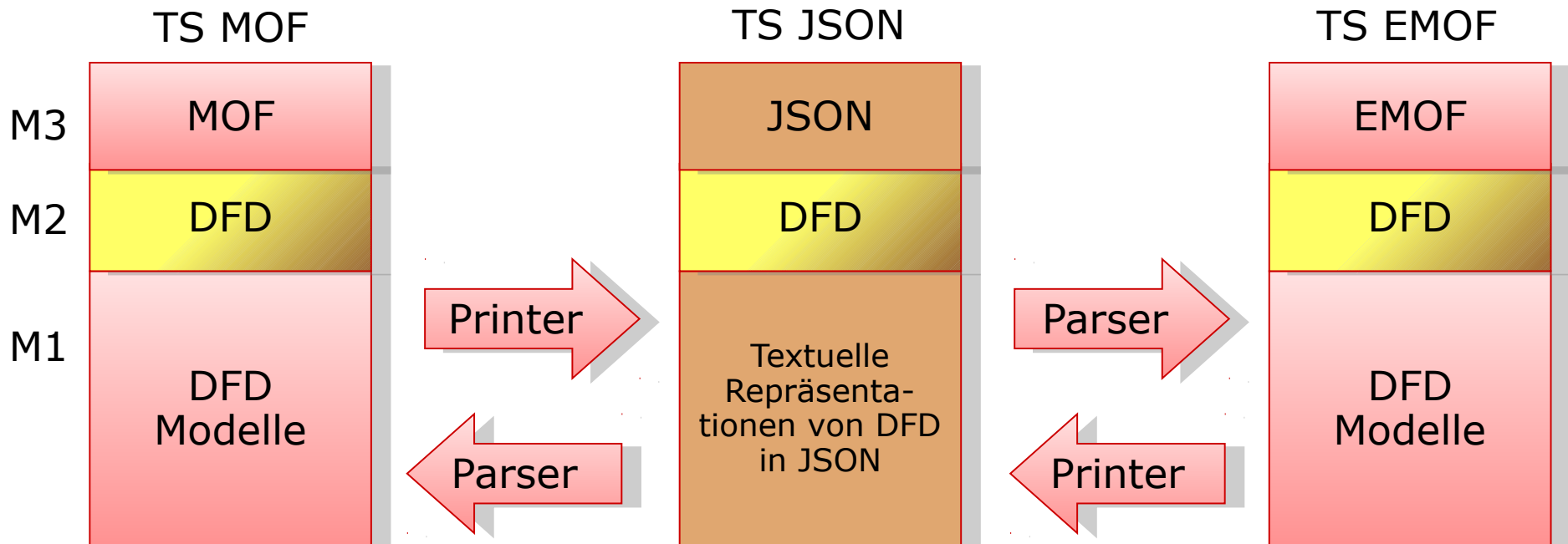
Ontology Technical Space (Reasoning World)



Transformative TS-Brücken mit JSON

31

- ▶ JSON (Java Script Object Notation) hat sich als beliebtes, einfaches Datenaustauschformat für Bäume etabliert <http://www.json.org/>
<http://www.ietf.org/rfc/rfc4627.txt?number=4627>
- ▶ i.W. ähnelt es XML, ist aber leichter lesbar



30.4 Architektur von SEU

32

Eine Softwareentwicklungs-Umgebung ist ein
Werkzeugsatz mit Daten, Steuer-, Prozess, und
Benutzerschnittstellen-Integration.

Architektur einer SEU

33



Schicht 3

Gemeinsame Benutzungs-Schnittstelle

Schicht 2

**Einzel-
Werkzeuge**

ToolServices

Kommunikations-
schnittstelle
zwischen Tools

Schicht 1

**Front End
Werkzeuge**

**Repository
(Interne Datenbasis)**

**Back End
Werkzeuge**

Tool-Export-Im-
port-Schnittstelle

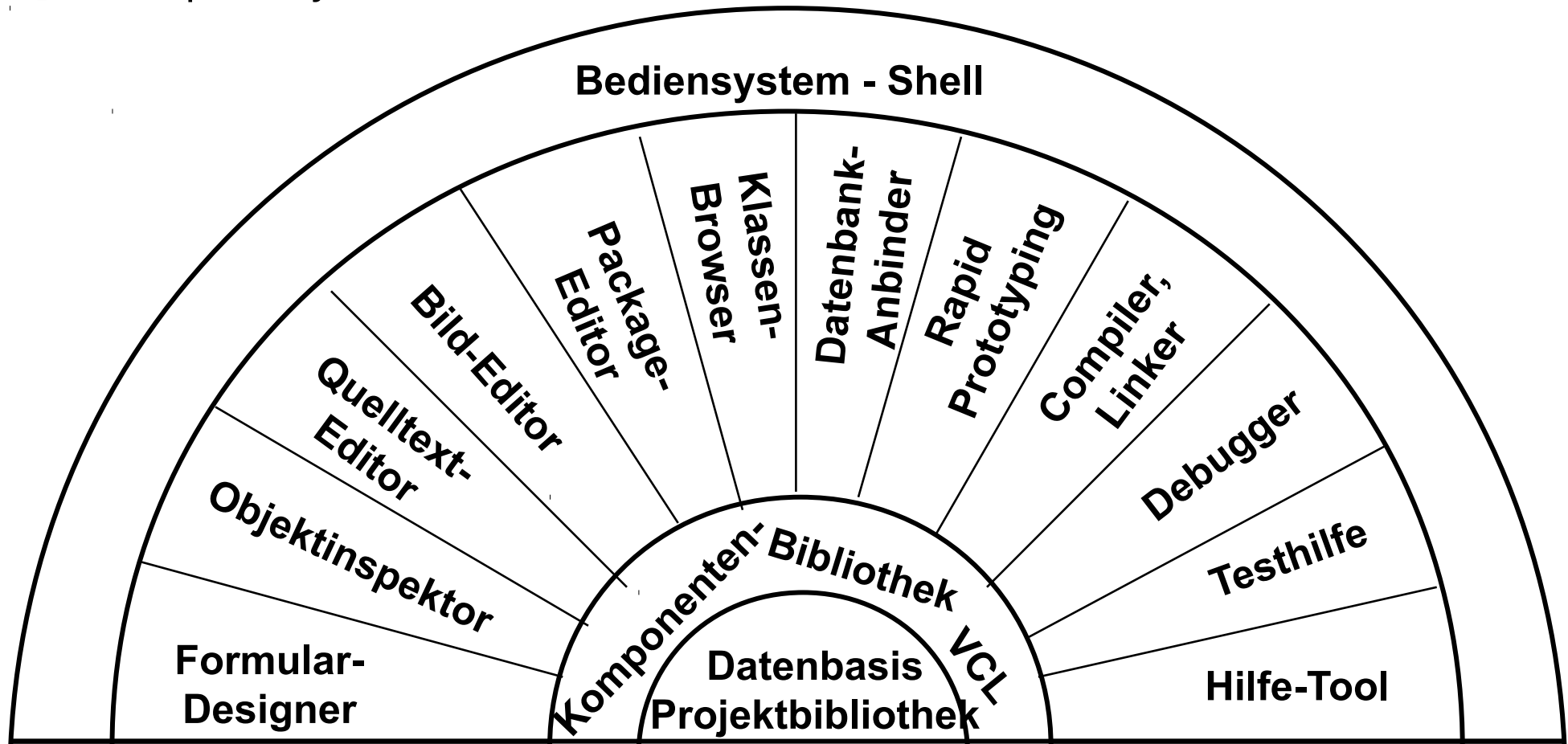
Schicht 0

**Externe Datenbasis
(externes Repository)
(Datenbank Dateisystem Web)**

Beispiel: Das Schalenmodell von Delphi

34

- ▶ Repository-basiert



UNIX Programmers Workbench (PWB): Stream- and File-Based

35

- ▶ Bell Labs developed a stream-based programmers' workbench on UNIX kernel. (1976)
 - UNIX had introduced the file system and streams (for C programs and shell scripts)
- ▶ http://en.wikipedia.org/wiki/Programmer%27s_Workbench_UNIX
- ▶ CACM publication:
 - <http://delivery.acm.org/10.1145/360000/359856/p746-ivie.pdf?key1=359856&key2=5161309211&coll=GUIDE&dl=GUIDE&CFID=55168257&CFTOKEN=9543918>
- ▶ “Notable firsts in PWB include:
 - The Source Code Control System, the first revision control system, written by Marc J. Rochkind
 - The remote job entry batch-submission system
 - The PWB shell, written by John R. Mashey, which preceded Steve Bourne's Bourne shell
 - The restricted shell (rsh), an option of the PWB shell
 - The troff -mm (memorandum) macro package, written by John R. Mashey and Dale W. Smith
 - The make utility for build automation
 - Utilities like find, cpio, expr, all three written by Dick Haight, xargs, egrep and fgrep
 - yacc and lex, which, though not written specifically for PWB, were available outside of Bell Labs for the first time in the PWB distribution”

30.5 Beispiel einer Referenzarchitektur für Werkzeug-Umgebungen: Das ECMA Referenzmodell für SEU

36

.. Der ECMA-Toaster

Standardisierungsorganisation

European Computer Manufacturing Association (ECMA)

37

- ▶ 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 von Kommunikationstechnologien (ICT) und Verbraucherelektronik (CER).
 - korrekten Gebrauch von Standards anregen und kontrollieren.
 - Veröffentlichung von 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

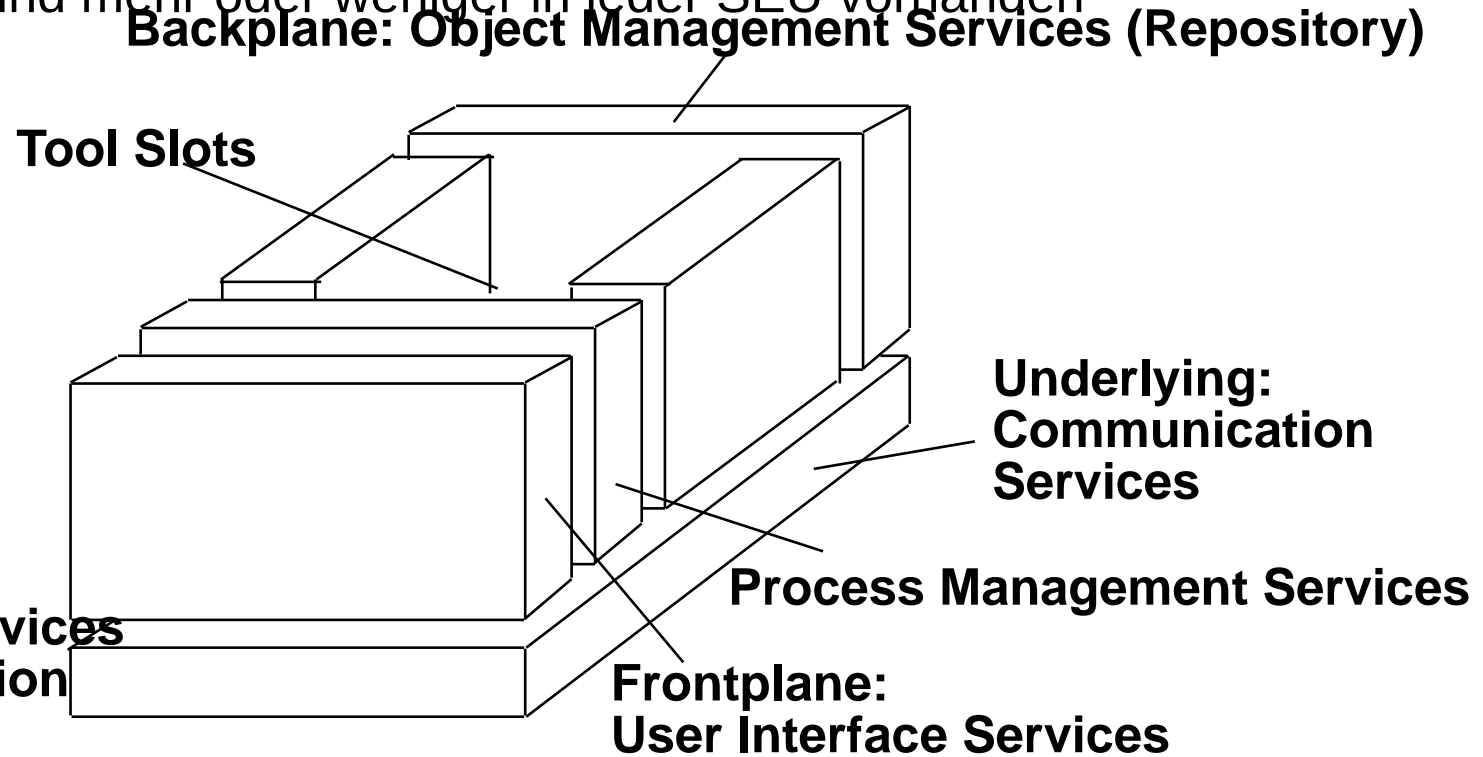
Quelle: <http://www.ecma-international.org/>



ECMA-Referenzmodell („ECMA Toaster“)

38

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



Sichten auf Dienste des ECMA-Referenzmodells

39

externe Sicht	beschreibt die externe Nutzung des Dienstes durch andere Dienste bzw. durch Werkzeuge od. den Nutzer
konzeptionelle Sicht	beschreibt die Semantik (Funktionalität), ohne Implementierung oder Verfügbarkeit für den Nutzer zu beachten
interne Sicht	beschreibt die spezifische Implementierung (Betriebssystem, andere Tools) für die Dienstauführung
Sicht auf Operationen	führt die Menge von Operationen eines Dienstes auf, die zur Erreichung der Funktionalität (konzeptionelle Sicht) benötigt wird
Sicht auf Typen	beschreibt das Datenmodell des Dienstes einschließlich der Informationen über dieses Datenmodell (Metamodell)
Sicht auf Regeln	beschreibt Regelmengen, die mögliche Mengen 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

40

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

- ▶ **User Interface Metadata Service** dient der Definition, Steuerung und Handhabung von Schemata zur Unterstützung der Benutzungsschnittstelle
- ▶ **Session Service** gewährleistet volle Funktion, unabhängig von 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

41

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 Rolle 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 von Aktivitätsteilen für andere Aktivitäten fest
- ▶ **Process State Service** sammelt und wertet Ereignisse von Aktivitäten während ihrer Ausführung aus, die für die Koordination und spätere Entscheidungsplanung anderer Projektaktivitäten notwendig sind
- ▶ **Process Ressource Management Service** verwaltet das Festlegen von Ressourcen zur Ausführung definierter Prozesse für Werkzeuge und Nutzer

ECMA Werkzeugdienste (Tool Services)

42

- ▶ 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 Datenbasis (Repository)

Object Management Services

43

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

- ▶ 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 Objektinformationen fest.
 - **Mandatory Integrity Service** gestatten den Schutz von SEU-Objekten vor unauthorisierten Änderungen, z. B. Eintragung "read only" usw.
 - **Mandatory Conformity Service** überwacht alle Aktivitäten zur Einhaltung von 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 von Nutzern zum System

30.6 Ein Metamodellgesteuertes Framework zur Werkzeugintegration (PCTE)

46

Mit eigenem Technikraum und Metasprache PCTE-OBS

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

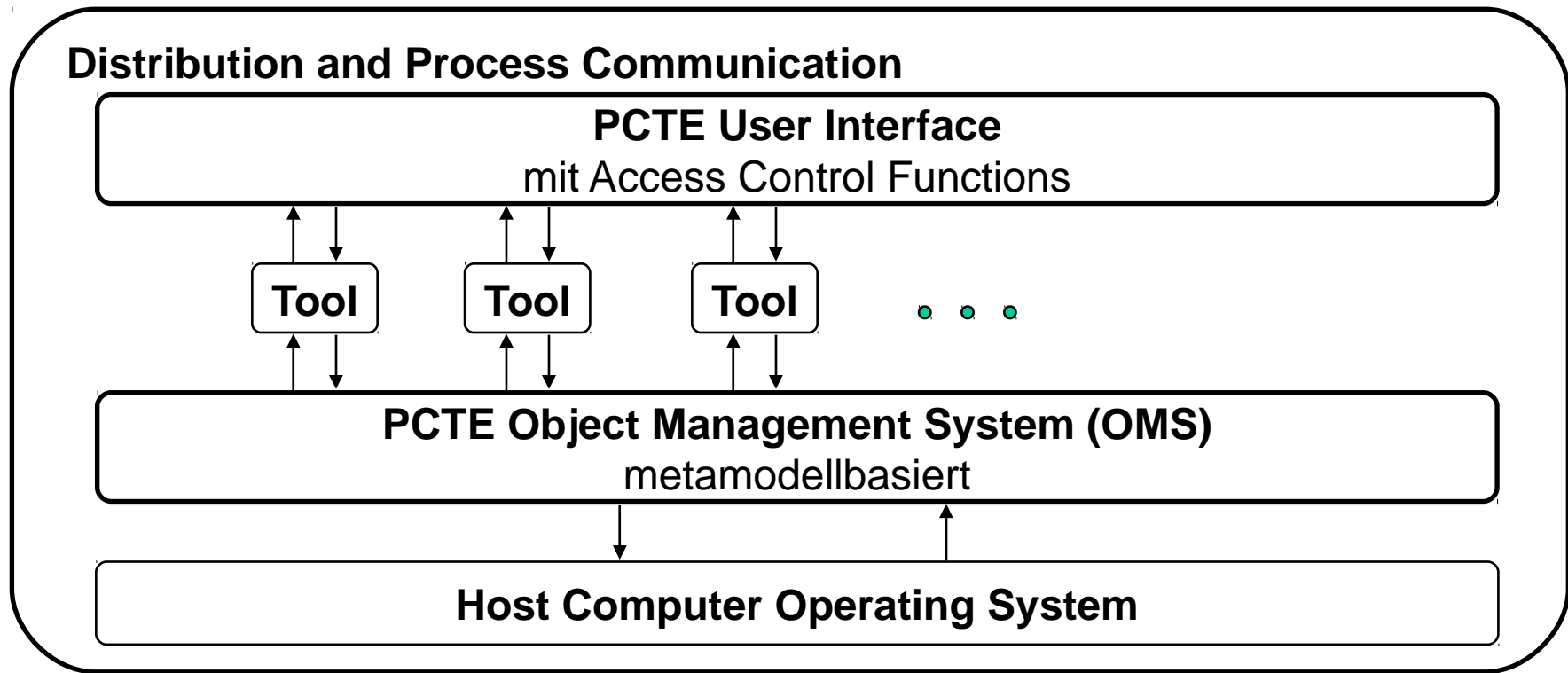
[http://citeseerx.ist.psu.edu/viewdoc/download?
doi=10.1.1.94.8315&rep=rep1&type=pdf](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.8315&rep=rep1&type=pdf)



Portable Common Tool Environment (PCTE+, HPCTE)

47

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



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 von 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 von Objekten sowie Schema-Management sind ebenfalls dezentralisiert.
- **erweiterbare Sicherheitsmerkmale**, wie Vertraulichkeit, geschützte Zugriffssteuerung für individuelle Objekte, Revisionsfähigkeit.

PCTE-OMS-Modell (Object Management System)

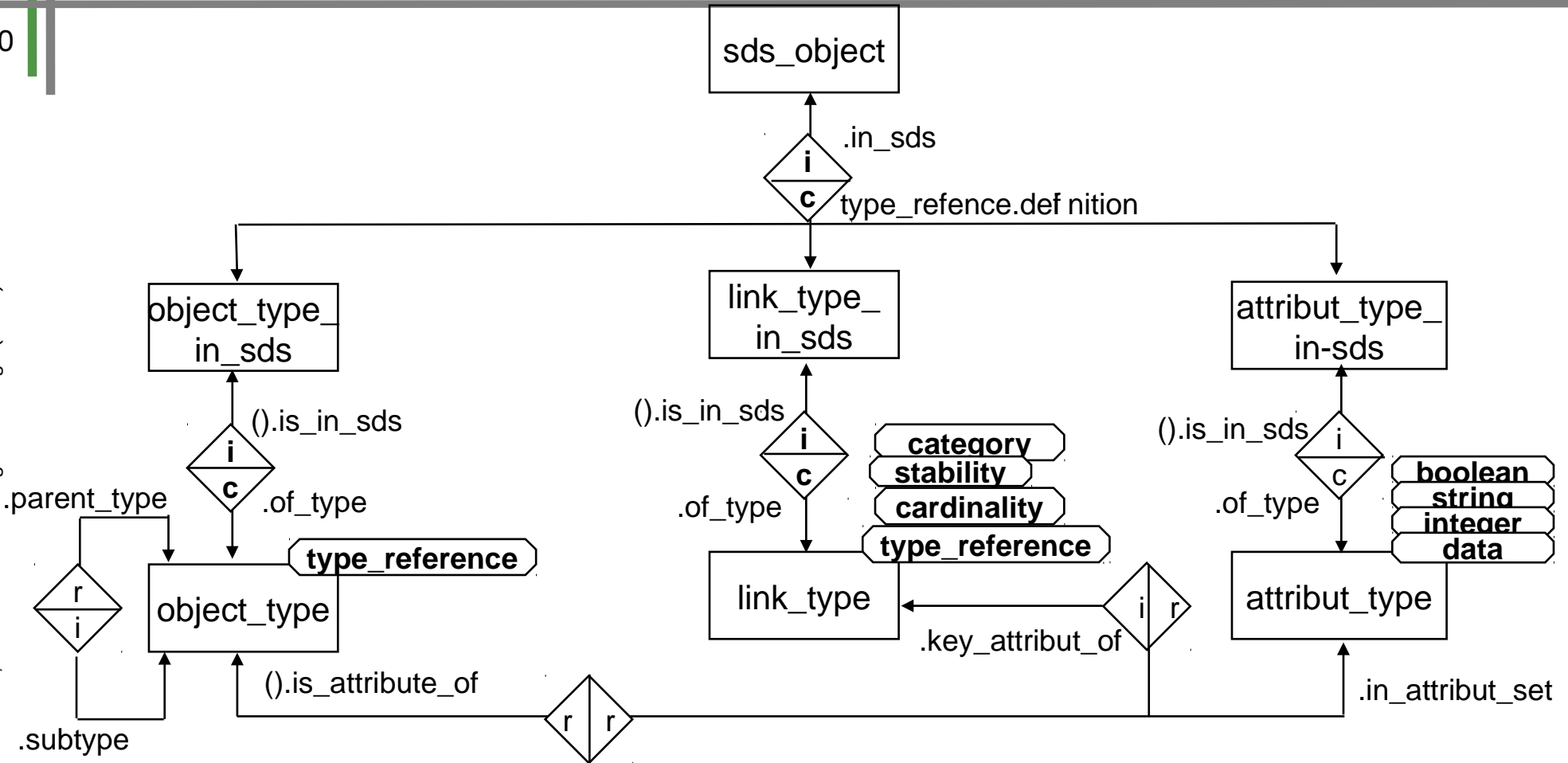
49

- ▶ Das OMS stellt Datentyp- und Datenspeichermöglichkeiten sowie Concurrency-Control-Mechanismen zur Verfügung,
 - definiert statische Informationen, die in der **object base** (Repository aller persistenten Daten) gehalten werden,
 - liefert Konzepte für die Softwareentwicklung, wie beispielsweise die (Typ-)Vererbung der objektorientierten Modelle.
- ▶ Das OMS ist metamodel-gesteuert. Es enthält **Typdefinitionen**, die in **Schema Definition Sets (SDS, Metamodellen)** beschrieben werden:
 - **Objekte**: Entitäten, auf denen Operationen der Werkzeuge ausgeführt werden. Instanzen können Dokumente, Textfiles, Quell- oder Objektcode, Task aber auch Geräte und Nutzer sein.
 - **Links (Assoziationen)**: gerichtete Beziehungen zwischen (Ursprungs-) und (Ziel-)Objekt (bidirektional)
 - **Attribute**: beschreiben 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 spezifiziert, DDL ist geliftet als Metasprache)

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

50

Prof. U. ARmann, Softwareentwicklungswerkzeuge (SEW)



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

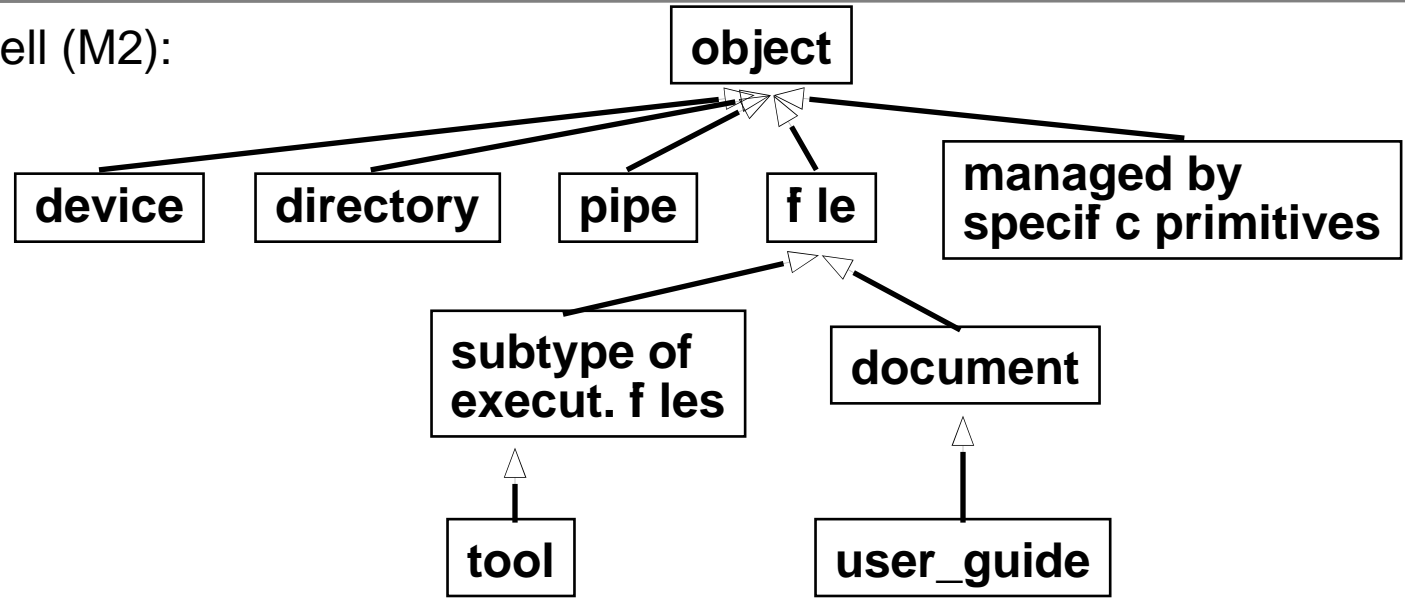
bei Anlegen eines neuen Objekts in Referenz zum existierenden.
 zwischen Ursprungs- und Zielobjekt
 kann z.B. reverse link für einen angelegten Link sein
 automatisch vom System (PCTE-OMS) gesetzt.



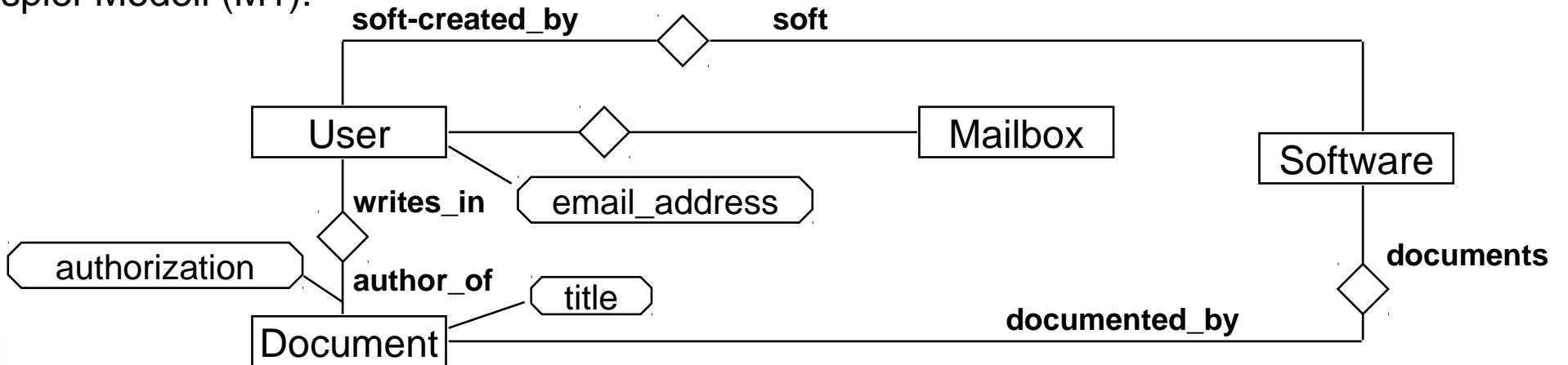
PCTE-Objekt-Strukturen mit erweitertem ERD

51

PCTE DDL Metamodell (M2):



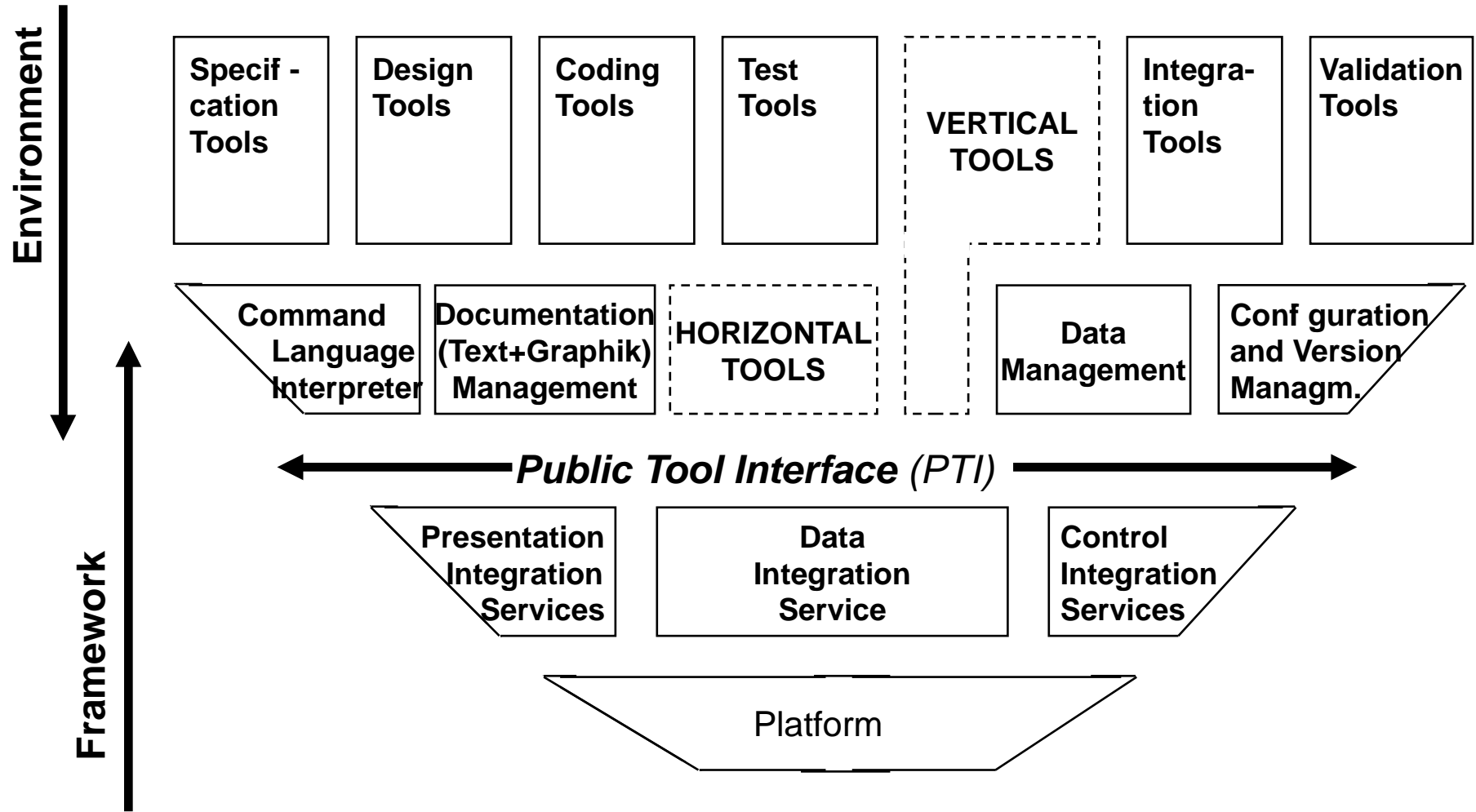
Beispiel-Modell (M1):



Emeraude PCTE Framework

52

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



More PCTE Implementations

53

- ▶ 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>

The End

54

- ▶ Werkzeuge werden unterschieden in
 - Repository-basierte
 - Datenfluss-gesteuerte
- ▶ SEU werden aus Werkzeugen komponiert
- ▶ SEU können Rahmenwerke (frameworks) zur Tool-Integration bereitstellen
 - ECMA hat die nötigen Dienste dazu definiert
- ▶ Datenintegration
 - Mit Strömen
 - Mit Datenaustausch
 - Mit Datenteilung (shared repositories)