31. Datenteilung von repository-basierten Werkzeugen (auf M1)

Prof. Dr. Uwe Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

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

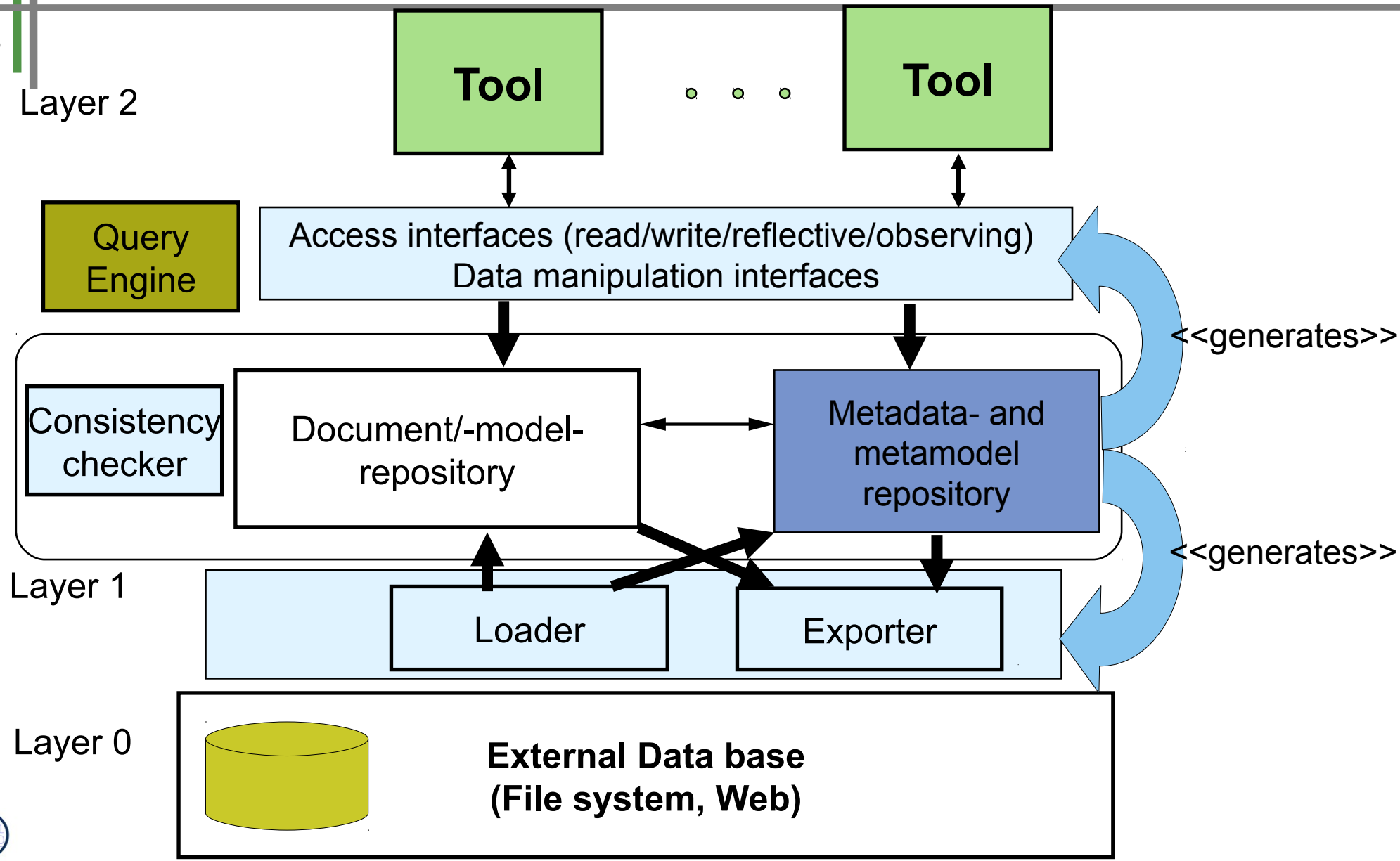
Version 12-1.3, 08.12.12

1) Komposition mit Werkzeug-Blutgruppen
    1) A-Werkzeuge
    2) E-Werkzeuge
    3) R-Werkzeuge
    4) M-Werkzeuge
2) Komponierbarkeit
3) Werkzeuge als Objekte (tool objects)
4) Einsatz des Graph-Logik-Isomorphismus

Literatur

- Johannes Siedersleben. Moderne Softwarearchitektur - Umsichtig planen, robust bauen mit Quasar, dpunkt-Verlag, 2004, Heidelberg
  - www.openquasar.de

Grobarchitektur bei Datenteilung (Metamodellgesteuertes Repositorium)



31.1 Effektkategorien für Werkzeuge (Blutgruppen) zur Komposition in SEU

Werkzeugkomposition und Datenteilung

- Realistische Werkzeuge verarbeiten immer eine DDL, DCL, DQL, DML, sowie Verhaltensspezifikationssprachen (BSL)
- Die individuelle Fertigung von solchen Werkzeugen ist zu teuer.

Wie kann man Werkzeuge einfach
zu grösseren Werkzeugen komponieren,
die ihre Daten teilen?

Wann können Werkzeuge parallel arbeiten?
Wann müssen sie synchronisiert werden?

Idee zur Lösung

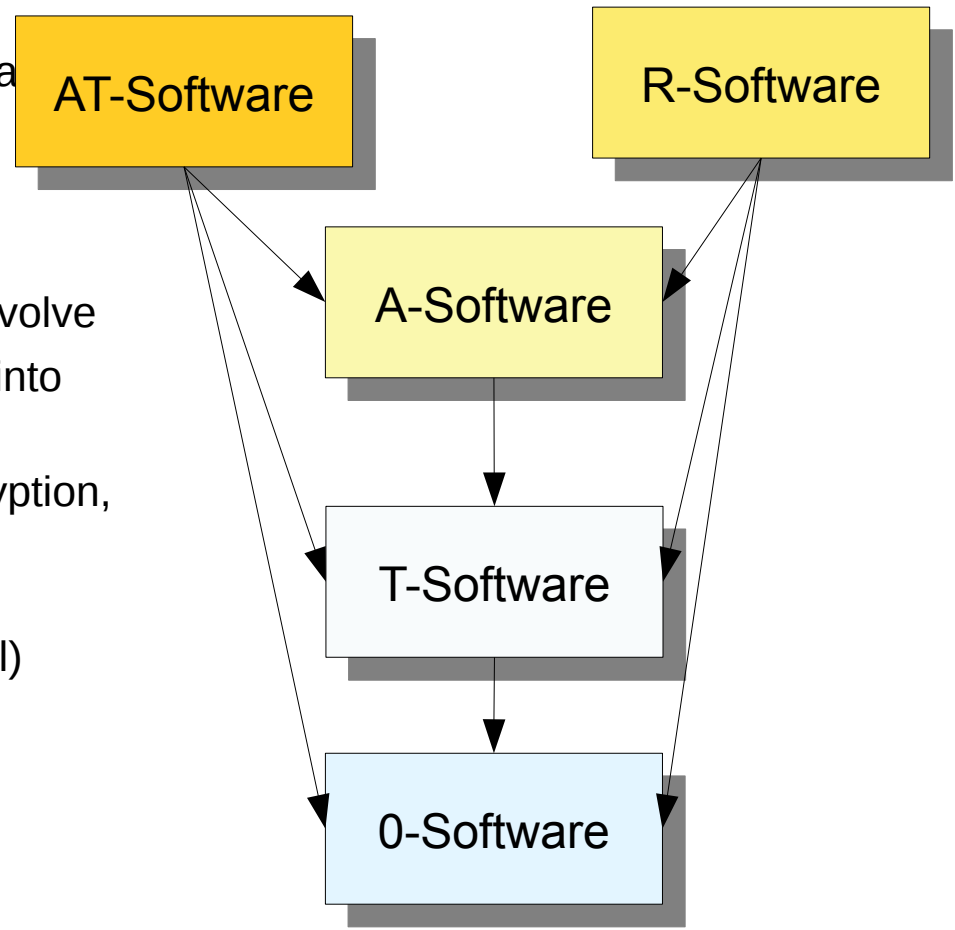
- [Siedersleben, Quasar]
- Betrachte zur Datenteilung (Datenintegration) von Werkzeugen seine Effekte eines Werkzeuges auf die Materialien im Repository (auf M1)
  - Komponiere die Werkzeuge, falls sie keine Konflikte im Zugriff auf das Repository haben

Idee: Übertrage Software-Blutgruppen-Gesetze auf Werkzeuge → Leser/Schreiber-Blutgruppen

Software Blood Groups (Blutgruppen) (Softwarekategorien nach Wiederverwendbarkeit)

- 0: independent of application and technology
  - JDK collections, C++ STL, GNU regexp
- A: application- or domain-related. Stems from domain model.
  - Client, Customer, ...
- T: technology-oriented APIs, independent of applica but not of technology
  - JDBC, CORBA CosNaming
- AT: depending on application and technology
  - To be avoided: hard to maintain, to reuse, to evolve
- R: for representation changes of business objects into external representations and back
  - Serialization, deserialization, encryption, decryption, packing, unpacking
  - Transporting an object from one language's representation to anothers (e.g., Java to Cobol)

- USES relationships:



Architectural Components

- 0-interfaces contain only technical types (strings, collections etc)
  - well reusable
- A-interfaces contain domain types (account, bill,..)
  - A-components live in the Application-Logic and the Database tier
  - Hard to reuse
- T-interfaces provide technical APIs
  - Necessary everywhere
- R-interfaces contain both, because they change representation
  - Are necessary in the middleware and data layer
  - Special kind of A
  - Can often be generated from specifications, hence not reusable, but re-generatable
    - XML tools, e.g., XMI (model interchange)
    - OMG MOF tools (Model-driven development)

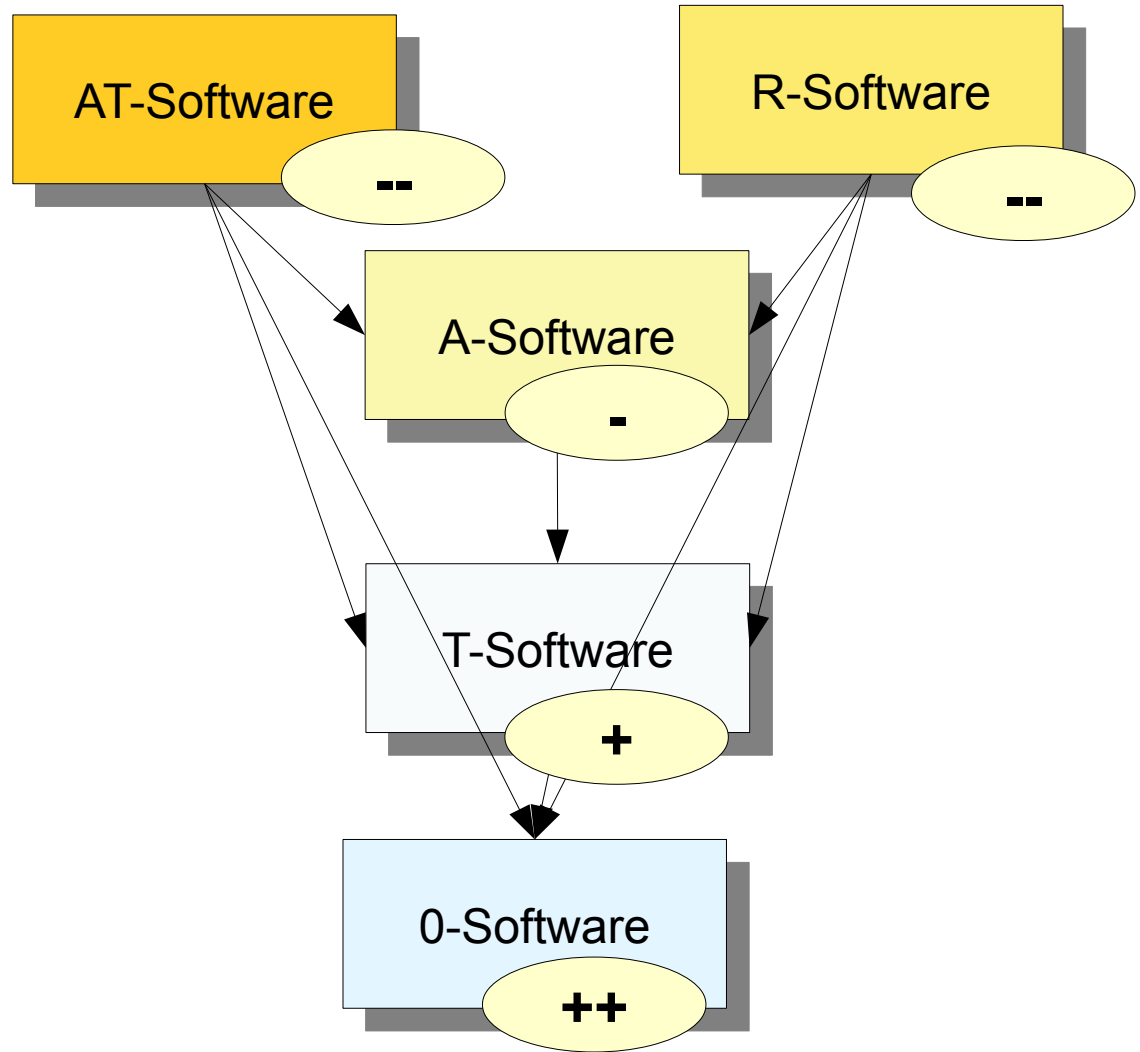
Zweck der Blutgruppen

- Zweck der Blutgruppen ist es, Anwendung und Technik möglichst weit voneinander zu trennen, um sie getrennt besser wiederverwenden zu können.

Siedersleben's Blutgruppen-Gesetz:
Jede Schnittstelle, Klasse, und Komponente gehört genau zu einer Softwarekategorie.

Wiederverwendbarkeit der Blutgruppen

- Die Wiederverwendbarkeit der Gruppen nimmt von 0 nach AT hin ab.
  - Technisch orientierte Komponenten sind leichter wiederzuverwenden
  - Anwendungsspezifische schwerer.
  - Problemfall AT
- Die Blutgruppen durchziehen alle Schichten der BCED-Architektur
  - Auf jeder Ebene gibt es Technik, Applikation, Repräsentation



CRUD-Schnittstelle von Materialien in Informationssystemen

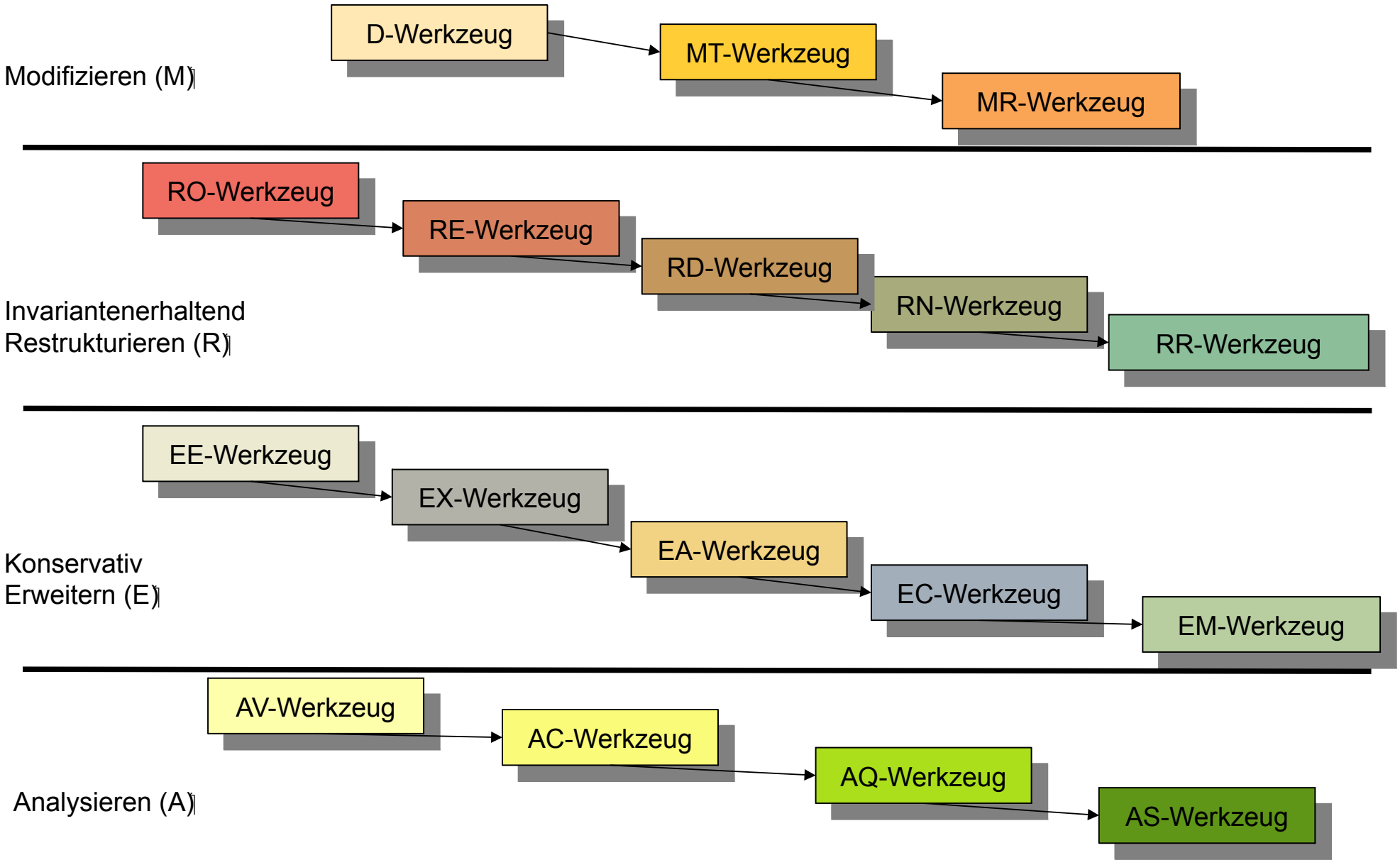
- Datenbanken kennen 4 elementare Operationen bzw. Effekte:
  - **C**reate: Anlegen eines Records/Tupels
  - **R**ead: Lesen eines Records
  - **U**pdate: Modifizieren eines Records
  - **D**elete: Löschen eines Records
- (Das Web kann als verteilte Datenablage angesehen werden)
- Wir kombinieren im Folgenden diese Effekte mit den Blutgruppen und verfeinern sie für Werkzeuge und ihre Materialien

	CRUD	http	webdav
Create	CREATE	POST	POST, MKCOL
Atomic Read	READ	GET, HEAD	GET, HEAD
Structural read			PROPFIND
Rearrange			MOVE, COPY
Write	UPDATE	PUT	PUT, LOCK, UNLOCK
Structural write			PROPPATCH
	DELETE	DELETE	DELETE

http://de.wikipedia.org/wiki/WebDAV

http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Effektkategorien für Werkzeuge (Wdh.)

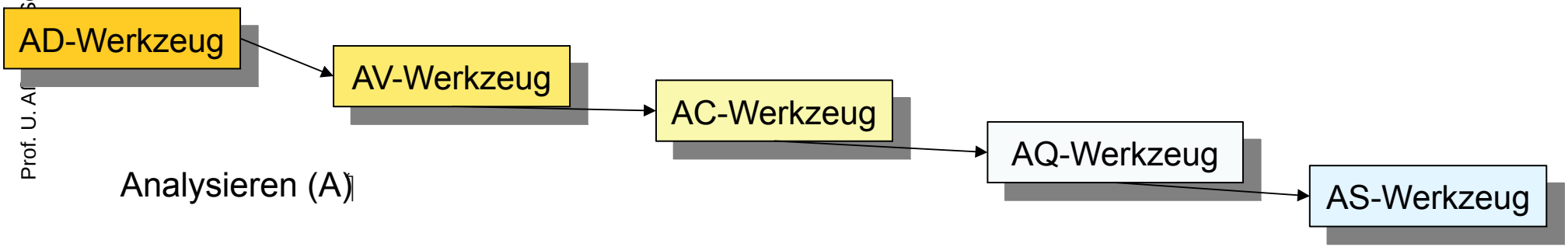


31.1.1 Einzelne Effektkategorien - Effektkategorie A (Analyse)

Effektkategorie Analyse-Werkzeuge (Blutgruppe A)

- Analysewerkzeuge lesen das Repositorium
  - verändern aber die Materialien nicht
  - nutzen Query-Sprachen (DQL, CQL)
  - alle Materialien bleiben unabhängig voneinander

- AS-Werkzeuge (Search, Automatisierungsgrade (1)-(3))
  - Suche von Informationen in einem Artefakt, seine Struktur, oder seine Attribute
  - Bsp.: Browser (Brauser), Lister, Editoren, Suchmasken-Programme oder Web-Browser.



AQ-Werkzeuge

- AQ-Werkzeuge (Query-Werkzeuge, query engines)
  - Abfrage komplexer Sachverhalte in Repositorien
  - Berechnung von Wissen, das implizit in den Artefakten vorhanden, aber nicht direkt ablesbar
  - Einsatz von Querysprachen
  - AQ-Werkzeuge benötigen AS-Funktionalität
- Beispiele:
  - logische Programmiersprachen
  - Querysprachen: Darauf aufbauende spezialisierte Code-Query-Werkzeuge wie Semmle/.QL bieten SQL-ähnliche Anfragesprachen an
  - Aufrufgraphvisualisierer
  - Metrikwerkzeuge
  - Informationssysteme für Geschäftsanalyse (business intelligence)

AC-Werkzeuge

- AC-Werkzeuge (syntactic checking, syntaktische Prüfung).
  - Prüfung von Konsistenzbedingungen über Repositorien (Integritätsregeln, integrity constraints, context conditions, validity checks).
    - Melden des Ergebnisses *(explanations)*
    - Vorschlag für Korrekturen (*correction hints*)
  - DCL und DQL werden eingesetzt
  - Syntaktische Integritätsregeln (kontextsensitive Bedingunge) lassen sich ohne Rückgriff auf die Semantik eines Programms überprüfen
  - Integritätsregeln der dynamischen Semantik sind Invarianten, die nur zur Laufzeit überprüft werden können
- Beispiele:
  - Parser, Typprüfer
  - Wohlgeformtheitsprüfer (constraint checker)
  - Stilprüfer

AV-Werkzeuge

- AV-Werkzeuge (Validation, semantic checking, semantische Prüfung).
  - AV-Werkzeuge prüfen die Semantik eines Programms.
  - Oft ist die semantische Prüfung in einen zweiten Technikraum ausgelagert
- Beispiele:
  - *Attribut-Evaluatoren* prüfen die statische Semantik eines Programms (Typregeln, Definitionsregeln, Sichtbarkeiten, etc.)
  - *Testwerkzeuge* führen eine semantische Prüfung durch, bei der validiert wird, ob eine neue Version korrekte Ausgabedaten erzeugt
  - *Theorembeweiser* beweisen, jenseits der Validierung durch Testen, die Korrektheit von Software.
    - Theorembeweiser können auch eingesetzt werden, um Verträge für Prozeduren und Methoden zu prüfen.
    - Auch können sie Anwendungen, die Frameworks benutzen, daraufhin überprüfen, ob sie das Framework korrekt benutzen.
  - *Modellprüfer (model checker)* prüfen, ob ein Beweis korrekt ist, bzw. ob ein System (Automat) ein Prädikat erfüllt

AD-Werkzeuge

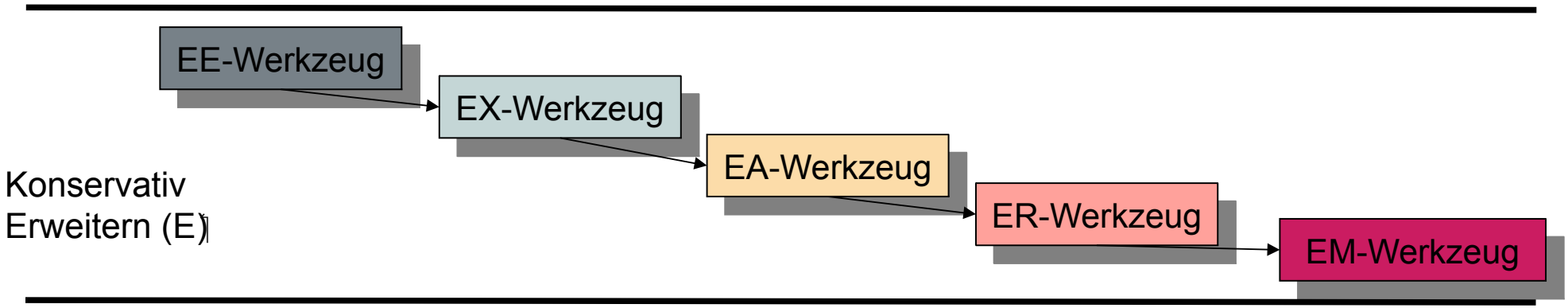
- AD-Werkzeuge (difference, Differenz- bzw. Ähnlichkeitsbestimmung)
  - Um zwei Artefakte eines Metamodells miteinander vergleichen zu können, müssen ihre Gemeinsamkeiten und Unterschiede bestimmt werden.
  - Artefakte können bezüglich eines Gleichheitskriteriums gleich, ähnlich oder völlig verschieden sein.
- Bespiele:
  - Differenzwerkzeuge für Code oder Modelle in Regressionstests
    - Prüfung von Ausgabedaten verschiedener Programmversionen auf Übereinstimmung
  - Differenzwerkzeuge für Dokumente

Zusammenfassung A-Werkzeuge

- Analysewerkzeuge arbeiten **idempotent**
  - Sie können immer wieder angewendet werden, ohne dass sich ihre Ergebnisse oder auch die Artefakte sich ändern.
- Einfache Datenintegration:
  - Analysewerkezeuge auf einfache Weise mit Werkzeugen, die andere Effekte auf das Material erzeugen, kombiniert werden.
- Insbesondere bedeutet das für die transaktionsorientierte Kopplung solcher Werkzeuge innerhalb von SEU, dass sie
  - als Leser und nicht als Schreiber gelten,
  - und somit parallel durchgeführt werden können.

31.1.2 Konservativ erweiternde Werkzeuge (E-Werkzeuge, Blutgruppe E)

- Konservativ erweiternde Werkzeuge (E-Werkzeuge) fügen den Materialien im Repositorium Informationen hinzu, zerstören aber keine Information.
- Dadurch sind sie von A-Werkzeugen unabhängig und einfach mit ihnen kombinierbar.



EM-Werkzeuge

- EM-Werkzeuge (Materialisierung) manifestieren Wissen, das schon implizit im Repositorium vorhanden war, in expliziter Form
  - Das ermittelte Wissen wird im Repositorium eingetragen
  - Dies kann nötig sein, um mit anderen Operationen fortfahren zu können
- Somit erhöhen EM-Werkzeuge die **Redundanz** des Repositoriums, verbessern aber den Zugriff auf die enthaltene Information
- Beispiele
  - Programmanalysewerkzeuge analysieren Programme, um Analyseinformation anzulegen und für andere Werkzeuge zur Verfügung zu stellen
  - Erstellung von Symbol- oder Definitionstabellen, die im Kontext vorhandene Definitonen von Namen zusammenfassen und zum Zugriff bereithalten
  - Ersetzungssysteme wie Termersetzungssysteme oder Graphersetzungssysteme können eingesetzt werden
  - Austauschformate in der Datenverbindung wie XMI

ER-Werkzeuge

- ER-Werkzeuge (Repräsentationswechsel, representation change)
  - Diese Art von Werkzeugen erzeugt für einzelne Artefakte oder sogar das ganze Repositorium eine weitere Repräsentation, ohne die existierenden Daten zu verändern.
  - ER nutzen dazu EM-Werkzeuge
  - Dies kann eine externe Repräsentation des Artefaktes sein, um den Artefakt zu serialisieren, zu ver- und entpacken, zu verschlüsseln oder in eine andere Sprache überführen.
- Beispiele
  - Transformatoren und Generatoren, die binäre Formen von Quelldateien erzeugen, indem sie nur die Repräsentation des Quellcodes verändern, aber die Semantik erhalten.
  - Werkzeuge für Austauschsyntax und Technikraumbrücken
  - Bauwerkzeuge (build tools) sind ebenfalls zu ER-Werkzeugen zu zählen, da sie den kompletten Aufbau der Systeme beschreiben und Eingabefiles für die Übersetzung in eine ausführbare Form liefern

EA-Werkzeuge

- EA-Werkzeuge (Adaption, Anpassung).
  - Oft besteht der Bedarf, einen Softwareartefakt auf bestimmte Randbedingungen anzupassen, z.B. auf die Plattform, auf der er ausgeführt werden soll, auf den Benutzer oder auf bestimmte funktionale Varianten hin.
  - Typisches Problem ist die Installation von Produkten, die sehr stark von der Umgebung des Systems abhängen und vielfältige Anpassungen erfordert.
- Beispiele
  - Installationswerkzeug
  - Interaktive Adaptionswerkzeuge (Assistent, wizard).

EX-Werkzeuge

- EX-Werkzeuge (Extension, Erweiterung)
  - EX-Werkzeuge bringen in ein Repositorium eine Erweiterung ein, ohne seinen Inhalt zu zerstören.
- Beispiele:
  - Annotationswerkzeuge für Modelle
  - Werkzeuge, die zusätzliche Sichten auf einen Artefakt einbringen, wie zum Beispiel aspektorienterten Weber
    - Trennung von Belangen (separation of concerns) in einen Kern und einen Aspekt
    - Ein Weber muss eingesetzt werden, der Kern und Aspekt vereinigt.
    - Für einzelne Webevorgänge kann das durchaus automatisch geschehen, für komplexere Prozesse wie z.B. die modellgetriebene Entwicklung jedoch ist das nicht unbedingt der Fall.

EE-Werkzeuge

- EE-Werkzeuge (Detaillierung, Elaboration)
  - Solche Werkzeuge werden dazu eingesetzt, um einen partiell spezifizierten Artefakt, z.B. ein Modell, zu vervollständigen. EE-Werkzeuge unterstützen dies durch die Präsentation der “Lücken” des Modells, d.h. durch die Präsentation von zu verfeinernden Teilen oder generischen Parametern.
  - Sie unterstützen also den Entwickler bei seinen Entscheidungen und sind somit Hilfswerkzeuge, die dem Entwickler bei der Vervollständigung seines Artefakts helfen.
- Beispiele:
  - Elaborations-Assistenten (refinement wizards), die dem Entwickler helfen, Parameter einzugeben, Masken, die noch fehlende Spezifikationsteile einfordern.
  - Zu dieser Kategorie zählen auch AC-Prüfwerkzeuge, die Inkonsistenzen entdecken und beim Entwickler fehlende Informationen abfragen.

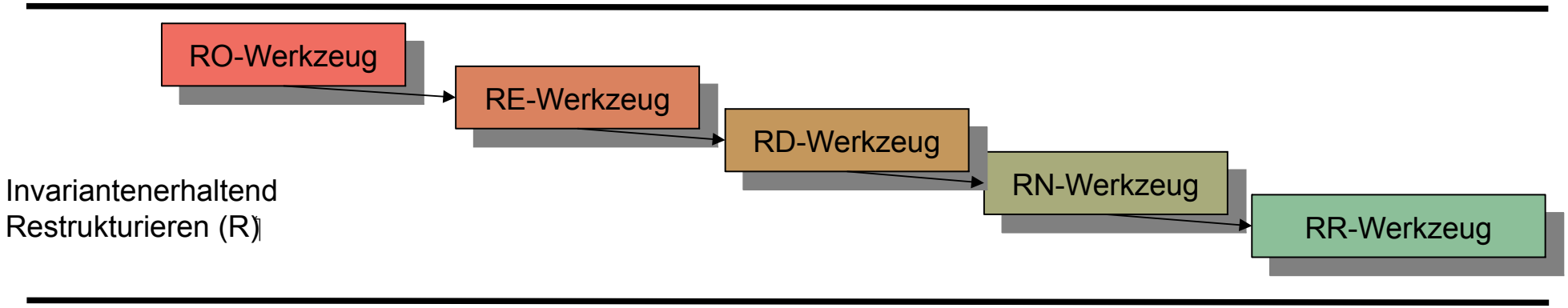
E-Werkzeuge

- E-Werkzeuge **verfetten** die Repräsentation des Artefakts, d.h. sie führen Redundanz ein.
- Konservativ erweiternde Werkzeuge sind ebenfalls idempotent, wenn sie die bei vorhergehener Ausführung erzeugten Informationen überschreiben.

- Beispiel:
  - Back-End eines Übersetzers: besteht aus EA-Werkzeugen, die für die Plattform, den Chip und die Laufzeitumgebung angepassten Code erzeugen, die Zwischenrepräsentation aber unangetastet lassen.

31.1.3 R-Werkzeuge (Blutgruppe R)

- Restrukturierende Werkzeuge verändern die Information im Repositorium, erhalten aber bestimmte Invarianten
  - Zum Beispiel die Semantik eines Artefakts
  - Dadurch bleiben die Ergebnisse von anderen Werkzeugen erhalten, insofern sie auf den Invarianten Informationen aufsetzen.
- Auch diese Werkzeuge können daher oft mit A- und E-Werkzeugen auf einfache Art kombiniert werden



RR-Werkzeuge

- RR-Werkzeuge (Refaktorisierung, refactoring)
  - Refaktorisierung meint, die Struktur eines Artefakts oder sogar des ganzen Repositoriums zu ändern, ohne die Semantik anzutasten.
  - Für den Einsatz eines RR-Werkzeugs bedeutet dies, dass es die Semantik des Artefakts im Repositorium invariant lässt, obwohl es an seiner Struktur Veränderungen vornimmt.
- Beispiele:
  - Ein RR-Werkzeug Namen von Variablen, Klassen oder Methoden verändern; so lange es das konsistent tut, bleibt die Programmsemantik erhalten.
  - In den letzten 10 Jahren haben Refakturierungswerkzeuge große Bedeutung gewonnen; mittlerweile gibt es keine Entwicklungsumgebung mehr, die keine Funktionalität für Umbenennung, Verschieben von Methoden, Aufsplitten von Klassen, etc. bietet.

RN-Werkzeuge (Normalisierung)

- RN-Werkzeuge restrukturiert die Datenbasis, um sie in eine Normalform zu bringen (spezielle Refaktorisierung).
  - Dabei kann Redundanz eingeführt oder auch eliminiert werden; entscheidend ist, dass der Endzustand eine normalisierte Form besitzt, die für zukünftige Operationen leichter auszuwerten ist.
  - Mit einer normalisierten Darstellung kann z.B. die Zahl der Fälle reduziert werden, die ein Folgewerkzeug zu untersuchen hat (vereinfachte Fallanalyse), oder es kann einfacher sein, einen Repräsentationswechsel durchzuführen.
- Beispiele:
  - Normalisierungen werden häufig in einem Übersetzer oder Refakturierungswerkzeug angewandt, z.B. werden Mehrfachzuweisungen in mehrere Einfachzuweisungen umgewandelt, abkürzte Pfade entfaltet
  - Normalisierungswerkzeug in einer Datenbank
  - Normalisierungswerkzeug in einem MDM

RD-Werkzeuge

- RD-Werkzeuge (Reduktion, vertikale Transformation)
  - Werkzeuge dieser Kategorie normalisieren in spezieller Weise, denn sie wandeln Schritt für Schritt ein Programm in einer höheren Sprache in eine Programm einer niederen Sprache.
  - Im Gegensatz zum allgemeinen Fall des Übersetzens wandelt dabei ein Reduktionsschritt genau ein Konstrukt der höheren Sprache in die niedere Sprache um (Flachklopfen, lowering). Damit werden RD-Werkzeuge oft als Präprozessoren eingesetzt.
- Beispiele:
  - Codegeneratoren für domänenspezifische Programmiersprachen (DSL). Diese wandeln ein Programm nicht direkt in Maschinensprache, sondern in eine Hochsprache um.
  - Content-Management-Systeme: Über mehrere Stufen werden die Dokumentformate in Web- und Grafikformate umgesetzt, um sie dem Benutzer anzeigen zu können. die dazu notwendigen Schritte sind oft Reduktionen.

RE-Werkzeuge

- RE-Werkzeuge (Elimination, Entfettung, Verschlankung)
  - RE-Werkzeuge bilden den Gegenpart zu EM-Werkzeugen.
  - Sie suchen nach redundanten Teilen der Datenbasis, die aus anderen rekonstruiert werden können und eliminieren diese (Verschlankung).
- Beispiele
  - Entfernen von abgeleiteten Relationen
  - Elimination gemeinsamer Teilausdrücke
  - Ausfaktorisieren von Prozeduren oder Oberklassen.
- RE-Werkzeuge sind auf AS-Werkzeuge zurückführbar, da
  - Redundanzen nur dann gefunden werden können, wenn Muster mehrfach im Artefakt gefunden werden können (Mustersuche nach Redundanzen).
  - Zusätzlich muß dann natürlich das Replikat eliminiert werden, was einer einfachen Transformation entspricht.

RO-Werkzeuge

- RO-Werkzeuge (Optimierung)
  - Optimierende Werkeuge transformieren den Artefakt bezüglich einer Optimalitätsfunktion, z. B. einer Kosten-Nutzen-Zielfunktion, die auf einem Kosten-Nutzen-Modell beruht.
  - Dabei bleibt die Semantik des Programms erhalten, es ändern sich aber seine Qualitätseigenschaften, z.B. der Verbrauch an Speicher, Rechenzeit oder an Energie.
- Beispiele:
  - Code-Motion-Optimierer
  - Code-Size-Optimierer
  - Register-Allokator

R-Werkzeuge

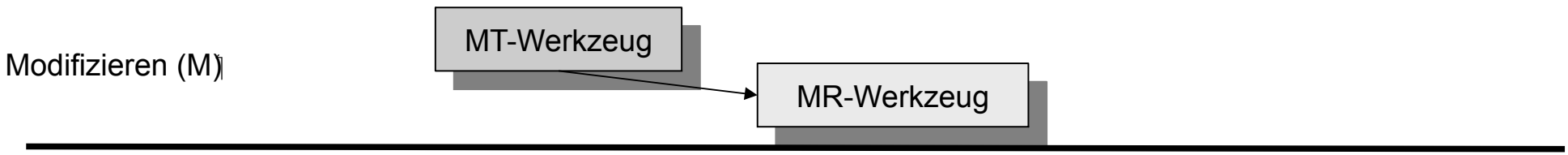
- R-Werkzeuge können nur dann mit A- und E-Werkzeugen kombiniert werden, wenn die Invarianten, die sie erhalten, mit dessen Funktionalität harmonieren.
- R-Werkzeuge sind kompatibel zu A- und E-Werkzeugen, falls jene auf Teilen des Repositoriums arbeiten, die nicht von der Invariante erfasst sind.

R-Werkzeuge - Beispiele

- Frontend eines Übersetzers: besteht aus RR-, EM-, und EX-Werkzeugen.
- Lexikalische Analysator:
  - wandelt die Zeichen der Eingabedatei in einen Strom von Lexemen (token), die dann vom
- Zerteiler: in einen Baum verwandelt werden, der der abstrakten Grammatik der Programmiersprache entspricht (RR).
  - Eigentlich müsste der Parser die Kontextbedingungen des Programms mitprüfen, da aber die Analyse von kontextsensitiven Grammatiken mindestens quadratischen Aufwand besitzt, spaltet man die Analyse der Namen, Typen, und Werte in eine separate AC-Phase ab (statische semantische Analyse).
  - Dabei zeigt jede Fehlermeldung der semantischen Analyse eine Verletzung der Kontextbedingungen des untersuchten Programms an, z.B. das Fehlen der Definition einer Variablen.
- Statische Semantische Analyse: Manifestationen von Namensrelationen in Definitionstabellen (EM-Werkzeug).
- Optimierer: enthält
  - RE-Werkzeuge (Entfernung von gemeinsamen Teilausdrücken in Grundblöcken),
  - normalisierende Vereinfachungen (RN-Operationen)
  - und RO-Werkzeuge (Operatorvereinfachung oder Befehlsanordnung).

31.1.4 Modifizierende Werkzeuge (M-Werkzeuge, Blutgruppe M)

- Modifizierende Werkzeuge ändern das Repositorium in beliebiger Art und Weise, sodass alle anderen Werkzeuge mit ihnen interagieren und alle Artefakte, die sie bearbeiten, voneinander abhängig werden.



MR-Werkzeuge

- MR-Werkzeuge (Reparatur, Automatisierungsgrad (4))
  - MR-Werkzeuge ermöglichen die Beseitigung von inkonsistenten Zuständen eines Softwareartefakts.
  - Angewandt nach AC oder AV-Werkzeugen
  - Hat man mit Hilfe von Prüfwerkzeugen der Kategorie AC oder AV Inkonsistenzen festgestellt, muß entweder durch Handprogrammierung oder mit Hilfe von geeigneten Werkzeugen Abhilfe geschaffen werden.
- Beispiel:
  - Versionsmanagementsysteme, die nebenläufige, in Konflikt stehende Änderungen an einem Produkt miteinander abstimmen und beseitigen können

MT-Werkzeuge (allgemeine Transformation)

- MT-Werkzeuge transformierent einen zugrundeliegenden Artefakt in beliebiger Weise in eine andere Form
  - Es werden keinerlei Invarianten erhalten, sondern der Artefakt wird erweitert, restrukturiert, verändert, sodass alte und neue Version sich stark unterscheiden können
- Beispiele:
  - Werkzeuge, die mit Term- oder Graphersetzungssystemen spezifiziert sind (Xcerpt, Xquery, Fujaba)

MD-Werkzeuge (Delete)

- MD-Werkzeuge löschen ein Material, so dass es nicht mehr zugegriffen werden kann.
- Beispiele:
  - Deallokatoren

M-Werkzeuge

- M-Werkzeuge sind am schwierigsten zu komponieren, da sie auf dem Repositorium vielfältige Abhängigkeiten zwischen den Artefakten erzeugen.

- Beispiele:
  - Übersetzer sollten, obwohl sie Programme transformieren, keine MT-Werkzeuge sein, da sie die Semantik des Quellprogramms im Zielprogramm wiederspiegeln müssen; sie sind daher RD-Werkzeuge.
  - Sie enthalten aber normalerweise RO-Werkzeuge, RN-Werkzeuge, und RE-Werkzeuge, sowohl im Back-End, dem Codegenerator, als auch im Middle-End, dem Optimierer.
  - Fehlerkorregierende Parser, die leichte Eingabefehler des Programmierers verbessern, entsprechen MR-Werkzeugen.

31.2 Komponierbarkeit von Werkzeugen auf M1-Repositorien

Werkzeug-Blutgruppen

- Die Effektkategorien der Werkzeuge bilden **Blutgruppen** für Werkzeuge [Siedersleben, Quasar]
- Die Blutgruppen drücken für verschiedene Klassen von Werkzeugen Kompatibilität bzw. Komponierbarkeit aus
- Die Matrix der Kompatibilitätsrelation drückt aus,
  - Welche Werkzeugklassen durch Komposition entstehen (Werkzeug-Infekt)
  - Wie sich Werkzeuge verschiedener Klassen anhand einer Kompatibilitätsrelation miteinander kombinieren (+, -, +-)

	Analyze	Extend	Rearrange	Modify
Analyze	A/+	E/+	R/+	M/+-
Extend		E/+	R/+-	M/+-
Rearrange			R/+-	M/-
Modify				M/-

Softwareinfektion und Werkzeuginfektion

- Die Regel des Softwareinfekts für Software-Blutgruppen gilt auch für Werkzeug-Effektkategorien (**Werkzeug-Infekt**) [Siedersleben Quasar]:

> Wenn ein repository-basiertes Werkzeug niederer Effekt-Kategorie
> ein Werkzeug höherer Kategorie aufruft,
> wird es selbst zu einem solchen.

- Beispiel:
  - ein A-Werkzeug, das Normalisierungen oder Reduktionen aufruft, wird selbst zu einem RN- oder RD-Werkzeug.
- Mit den Werkzeug-Blutgruppen können auch komposite Werkzeuge in einer SEU klassifiziert werden, in dem man sie auf die Kategorien der Teilwerkzeuge zurückführt.

Komponierbarkeit von Tools (Leser-Schreiber-Kompositionsgesetz)

Jedes Werkzeug besitzt eine Leser-Schreiber-Blutgruppe, anhand derer es mit anderen komponiert werden kann.

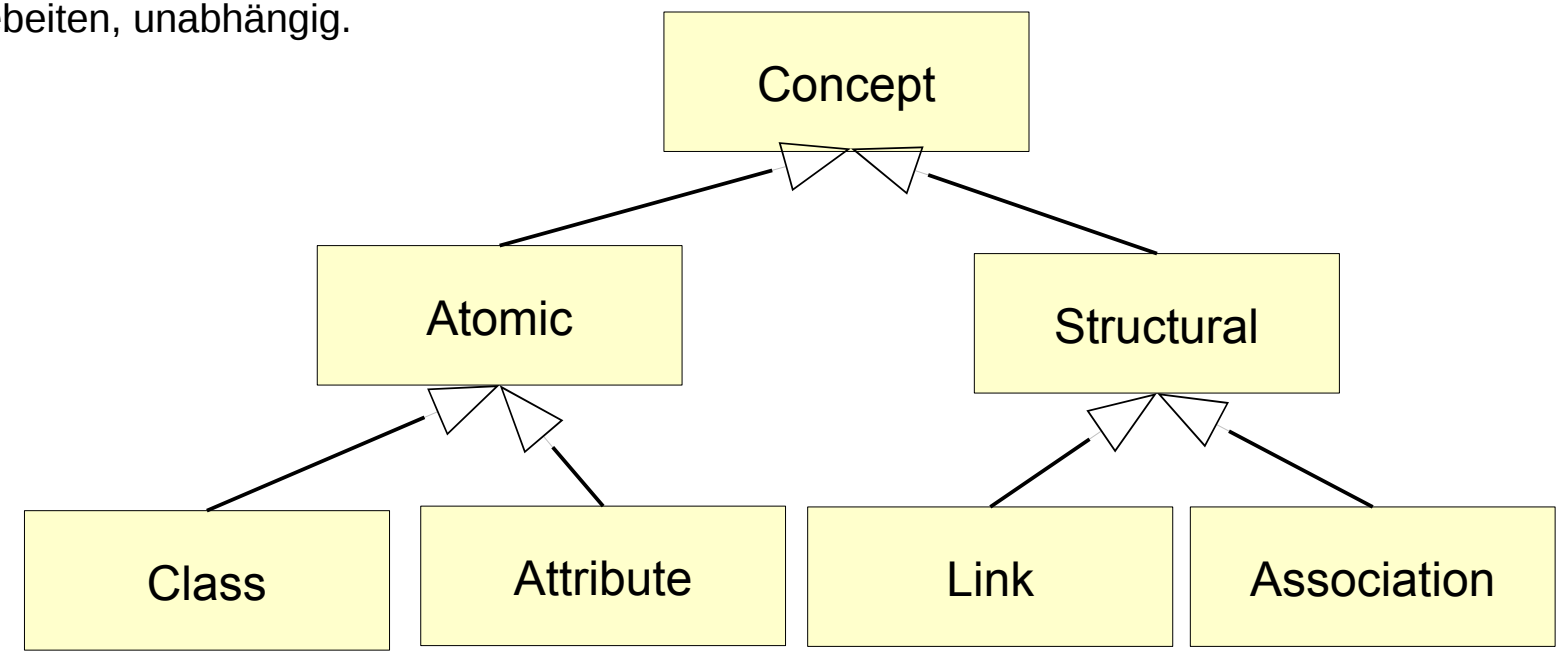
31.3 Datenteilung mit rollenbasiertem Entwurf

Ein Werkzeug (*tool*) ist ein Objekt, das einen Zustand (Rolle) über Material-Objekten in einem Repositorium verwaltet.

Siehe “Design Patterns and Frameworks”, Kapitel “Tools and Materials”

Ausnutzung von Struktur (Frames) als Eigenschaft der Metaklassen

- In einem Werkzeug können Informationen in Materialien auf zwei Arten gespeichert werden:
  - Atomar in einem Objekt (atomar im Rahmen, frame, framed representation)
  - In Relationen, durch Zeiger (structural)
- Alle Metaklassen von M3 bzw. der M2-DDL sind entweder *atomic* (ihre Instanzen werden atomar gespeichert) oder *structural* (Ihre instanzen werden strukturell gespeichert)
- Effekte auf Atomic beziehen sich auf nicht auf Structural und umgekehrt
- Werkzeuge, die auf Atomic Materials arbeiten, sind von Werkzeugen, die auf Structural Material arebeiten, unabhängig.

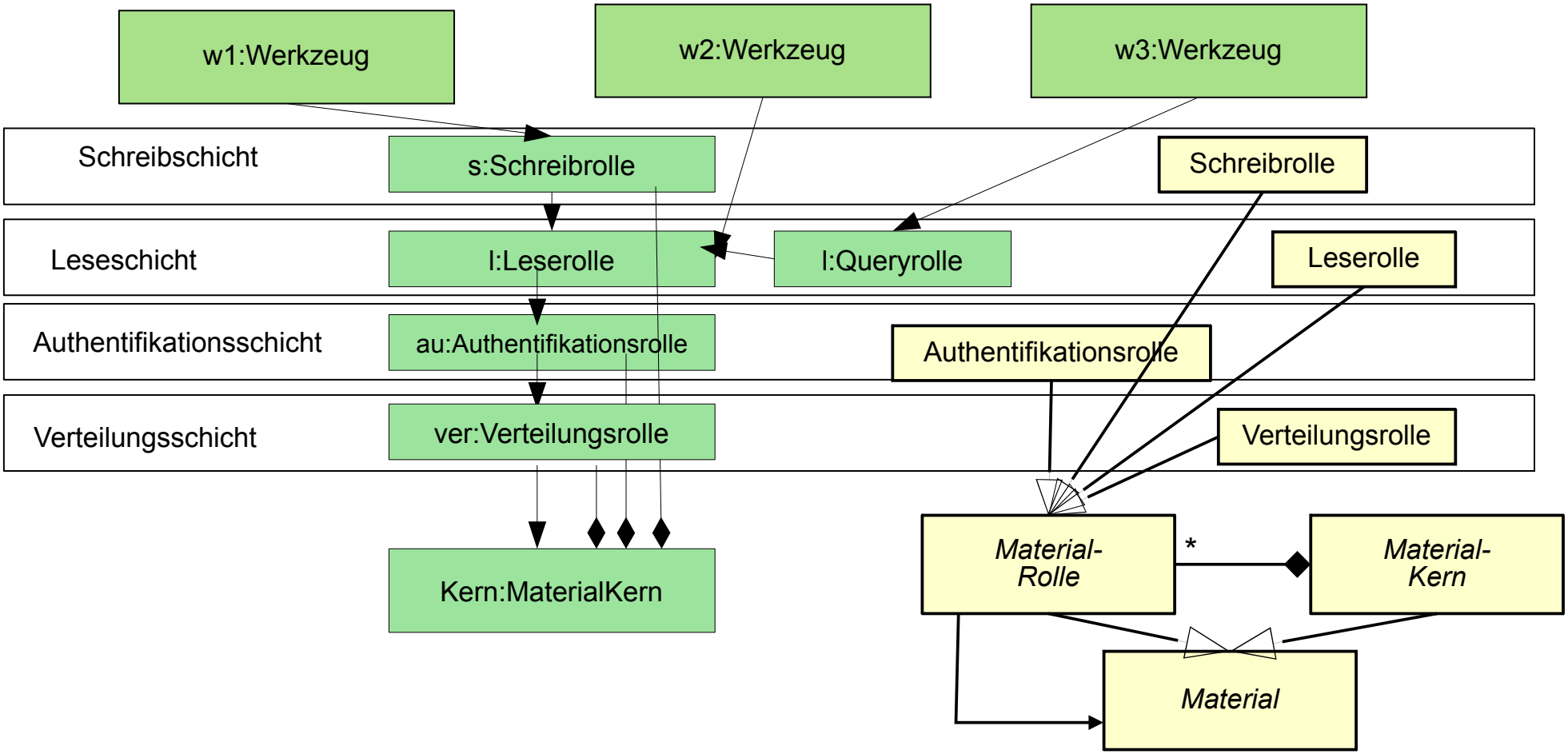


Werkzeug-Effekte auf Materialien und Ihre Prägung für Rollen

- CRUD wird im Folgenden um Strukturinformation erweitert
- **Erweiterungs-Rolle (Extend-Operation mit E-Effekt)**
  - 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
- **Restrukturierung (R-Effekt)**
  - **Atomare Restrukturierung** verändert den Zustand der atomaren Attribute des Objektes, z.B. vergibt neue Namen
  - **Strukturelle Restrukturierung** verändert die strukturellen Attribute des Objektes, d.h. des Objektnetzes, ohne die atomaren Attribute zu modifizieren
- **Modifikation (Update-Operation, M-Effekt)** wird aufgespalten in:
  - **Atomare Modifikation** ändert den Zustand der atomaren Attribute des Objektes
  - **Strukturelle Modifikation** ändert den Zustand des Objektnetzes, ohne die atomaren Attribute zu modifizieren

Kollaboration von Werkzeugen auf dem Repositorium mit Rollenobjekten (Wdh.)

- Ein gutes Entwurfsmuster für Repositorien und ihre Zugriffsschichten bietet das geschichtete *Role Object Pattern* von Bäumer et.al. (Siehe Kapitel in Design Patterns and Frameworks)
- Dekorator-Ketten regeln den Zugriff über Schichten



- Folgende Rollen können komponiert werden:

Lese-Rollen sind untereinander unabhängig

Struktur-Rollen und Atomare-Rollen sind voneinander unabhängig

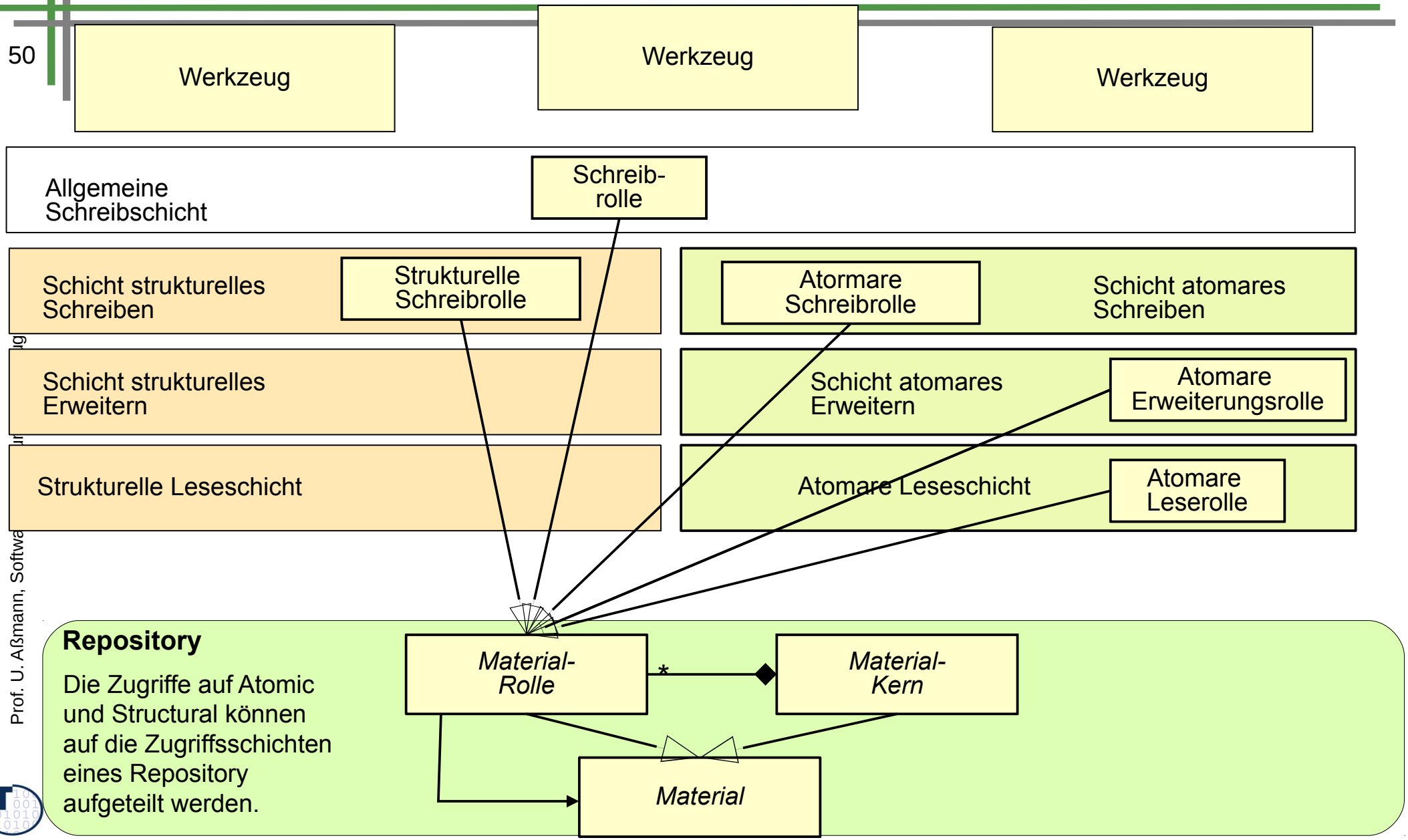
Erweiterungsrollen sind von Leserollen unabhängig

Erweitertes Interferenzgesetz

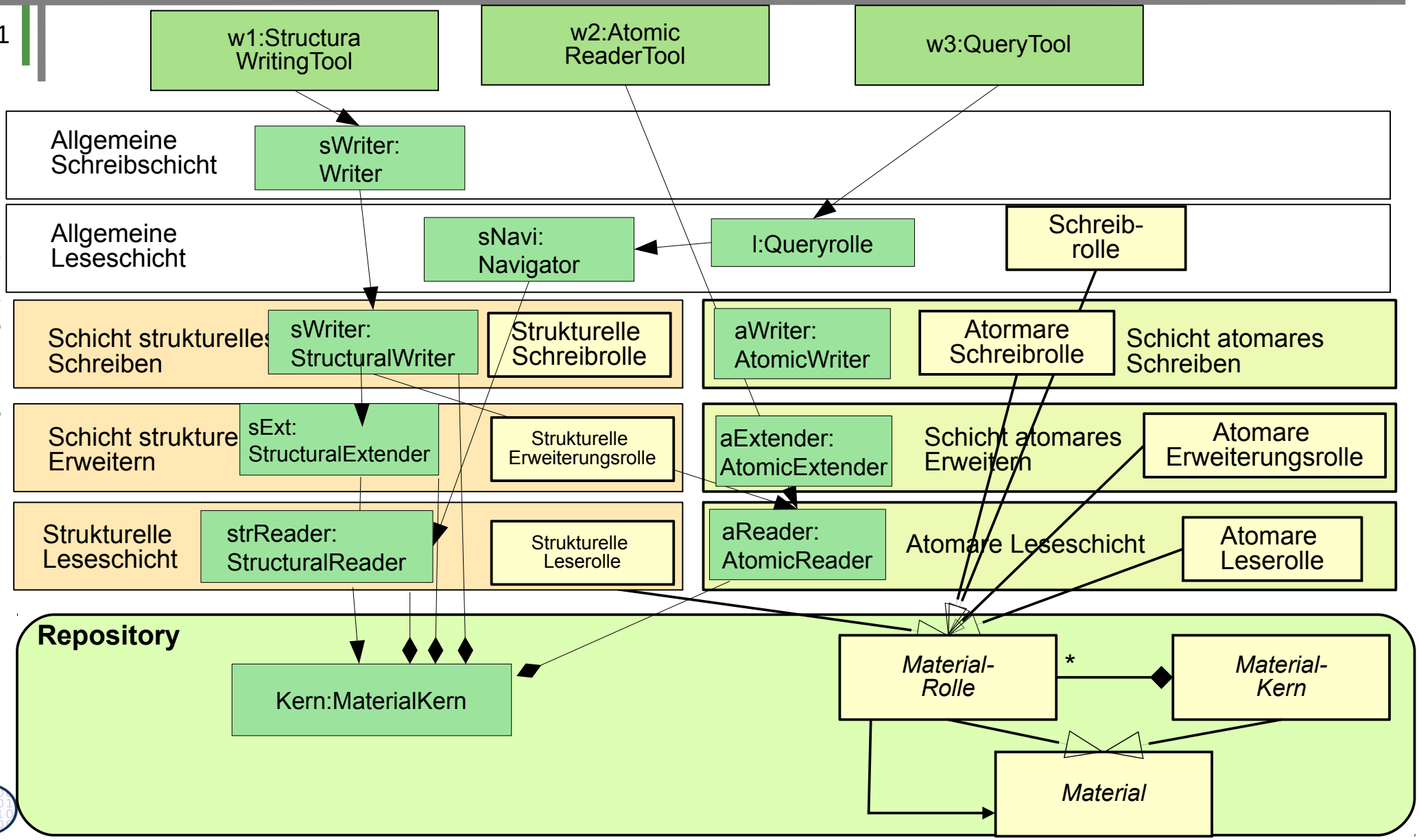
- Genauer drückt dies folgende Tabelle mit Interferenzen aus

		Analyze	Extend			Rearrange	Modify		
			atomic	structural	all		atomic	structural	all
Analyze		+	+	+	+	+	+-	+-	+-
Extend	atomic		-	+	+-	+		+	-
	structural		+	-	+-	+-	+	-	-
	all				+-	+-	-	-	-
Rearrange						+-	+	-	-
Modify	atomic						-	+	-
	structural							-	-
	all								-

Effekte auf Materialien und Ihre Prägung für Rollen

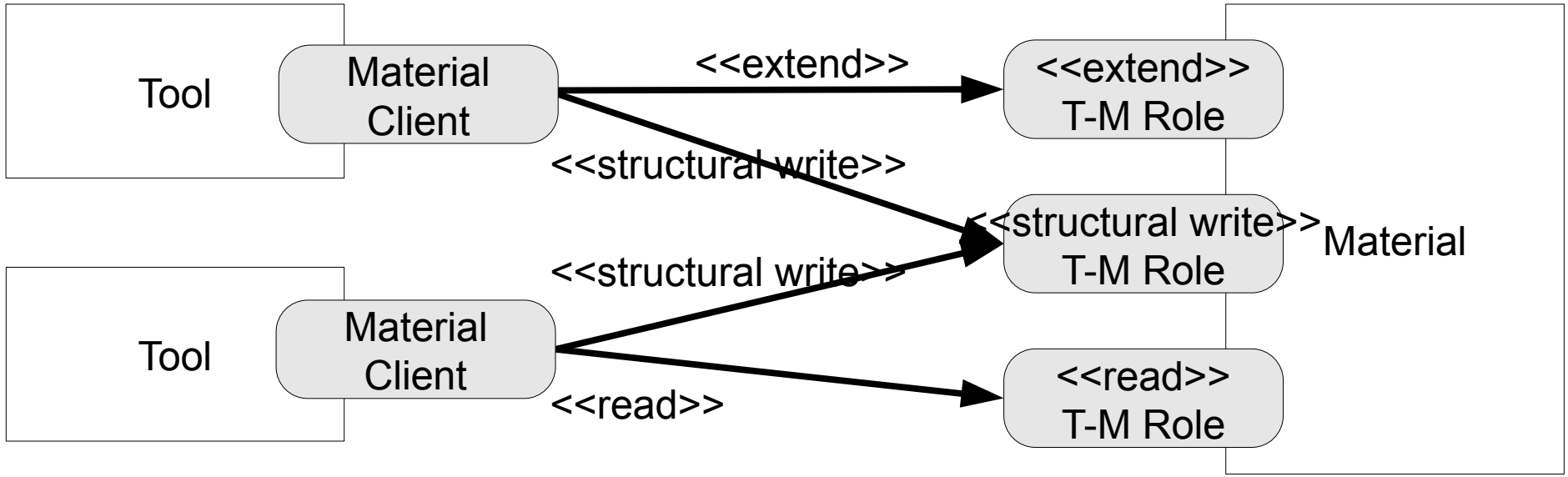


Collaboration of Tools on Repository
Shared Decorator-Chains for Shared Access



Leser-Schreiber-Kompositionsgesetz für Werkzeuge und Material-Zugriffsschichten

- Folgende Werkzeugobjekte (tool objects) können komponiert werden:
  - Leser (Leserollen nutzende Werkzeuge) sind untereinander unabhängig
  - Struktur-Werkzeuge und Atomare-Werkezeuge sind voneinander unabhängig
  - Erweiterungs-Werkzeuge sind voneinander unabhängig
- Andernfalls
  - Konkurrierende Schreiber und Leser müssen durch die Schreibschicht synchronisiert werden (Leser/Schreiber-Synchronisationsprotokoll)



31.4 Einsatz des Graph-Logik-Isomorphismus zur Konstruktion von lesenden und schreibenden Werkzeugen

Ein typisierter Multigraph ist äquivalent zu einer Logik-Faktenbasis erster Ordnung.

[Bruno Courcelle, Handbook of Theoretical Computer Science, 1990]

- Ein Artefakt (Modell, Dokument, Code) kann im Repositorium gespeichert sein als
  - Relation (Schema: Relationale Algebra; keine Objekte, nur Werte)
  - Bäume und Graph mit typisierten Knoten (Objekte) und Kanten (Schema ERD, UML-CD, MOF, EMOF)
  - Faktenbasis für eine Logikmaschine (Schema RDFS, F-Logic)
- Umwandlung zwischen diesen Formaten bzw. Uminterpretation der Formate auf Grund des Graph-Logik-Isomorphismus möglich

Wdh.: SEU mit Ersetzungs- und Logik-Werkzeugen

Spezial-
Werkzeuge

Blutgruppe A

Logik-
werkzeuge

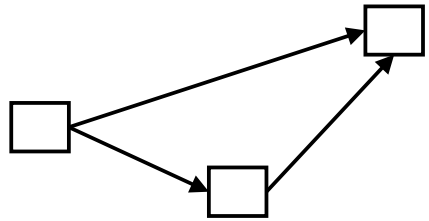
Blutgruppe M

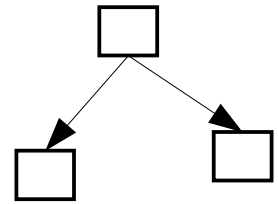
Graphersetzungs-
werkzeuge

Termersetzungs-
werkzeuge

Interpretation als Fakten

Bäume und
Graphen
Im Speicher





Persistente Bäume und Graphen

Nutzung beim Bau von A-Werkzeugen

- Queries:
  - Relationale Anfragesprachen verwendbar (siehe Semmle, SPARQL, oder SQL)
  - Datalog verwendbar (s. ST-II)
  - Kantenadditionssysteme (EARS) verwendbar (s. ST-II)
- Beispiele:
  - Analysierende Übersetzerphasen, wie die semantische Analyse oder die Programmanalyse.
    - Insbesondere bilden Konsistenzregeln und Kontextbedingungen der semantischen Analyse Integritätsbedingungen auf dem Programm.
    - Reengineering-Werkzeuge können mit Logik Informationen aus Altsystemen ziehen
  - Klassengraphen können als Faktenbasis einer Logikmaschine behandelt werden
    - Entwurfsmuster können als Logikregeln ausgedrückt werden
  - Konfigurationsmanagement kann als Deduktion über Konfigurationsgraphen ausgedrückt werden

Bau von EM-Werkzeugen mit Logik

- Um Erweiterungen von Graphen zu berechnen und zu materialisieren, kann ebenfalls Logik verwendet werden.
- Da Logik ohne Seiteneffekte auf die Datenbasis arbeitet, muss Information, die berechnet werden soll, mitgeschleift werden
  - Die Graphen, die materialisiert werden sollen, müssen, als Parameter durch Anfragen durchgeschleift und durch Suchergebnisse sukzessive erweitert werden,
  - während die Datenbasis unangetastet bleibt.
- Danach kann man die Ergebnisse der Anfrage materialisieren, d.h., der Faktenbasis zugeben.
- Das bedeutet, dass ein logikbasiertes E-Werkzeug, das einen Artefakt erweitern will, zunächst Graphen zunächst als Parameter von logischen Queries berechnen muss, und dann erst materialisieren kann.

Bau von RM und RC-Werkzeugen

- können mit Logik spezifiziert werden.
- Beispiel:
  - Generatoren, die ein Repositorium serialisieren, können durch Logik beschrieben werden.

Bau von M-Werkzeugen mit Logik: Schwierig

- Die Erstellung von weitergehende Transformationen ist mit Logik -- wegen ihrer Dekarativität - nur durch Wechsel des Repositoriums möglich,
  - wenn während der Abarbeitung daser gesamte transformierte Graph Repositoriums als Parameter der Query berechnet wird und man das Repositorium nach der Query komplett auf das Resultat umsetzt.
- Teuer: Da diese Operation relativ viel Speicher kostet, jedenfalls mehr als eine Veränderung des Repositoriums, wird in diesen Fällen Logik selten eingesetzt.

- Zur echten Transformation eines Artefakts können eingesetzt werden:
  - Metaprogramme
  - Ersetzungssysteme.
    - Termersetzungssysteme
    - Graphersetzungssysteme

Metaprogramme für M-Werkzeuge

- **Metaprogramme** sind Programme, die Programme erzeugen oder Transformieren.
  - Sie haben seit Lisp eine lange Tradition
- **Statische Metaprogramme** produzieren ein Programm, in dem sie selbst nicht enthalten sind.
  - Sie funktionieren also als reiner Code-Generator.
- **Dynamische Metaprogramme** laufen ständig mit der Anwendung mit und regenerieren Teile neu.
  - Dynamische Metaprogramme sind allerdings laufzeitintensiv und verhindern eine statische Analyse der dynamisch produzierten Programme.

Termersetzungssysteme

- Termersetzungssysteme oder XML-Ersetzungssysteme (Bsp. Xcerpt) setzen voraus, dass baumartige Datenstrukturen vorliegen und operieren daher meist auf dem abstrakten Syntaxbaum (AST).
- Einsatz:
  - Identifikation von Baummustern,
  - Verschlankungen
  - oder Normalisierungen.
- Strategien:
  - Freies Ersetzen (chaotic rewriting): alle Regeln solange angewendet werden, bis das Repositorium sich nicht mehr ändert
  - Spezialstrategien: Die Spezifikation von Strategien möglich, die die Regeln in bestimmter Reihenfolge auf das Repositorium anwenden
    - Beispiel: Das alternierende Suchen und Löschen von gefundenen Informationen.
  - Strategien sind also Regeln zweiter Ordnung, die Regelanwendungen steuern können

Graphersetzungssysteme

- Graphersetzungssysteme unterliegen der Einschränkung auf Baumdaten nicht.
  - Sie suchen und ersetzen Graphmuster in Graphen.
  - Graphersetzungssysteme beschreiben Strukturen, d.h. können für strukturelle Werkzeuge eingesetzt werden
- Beispiele:
  - Softwareentwicklungsumgebung (PROGRES) reduziert vielfältige Aufgaben im Entwicklungsprozess auf Graphersetzung (und Logik)
    - Übersetzung [Nagl]
    - Verfeinerung im Entwurf [Schürr, Lewerentz]
    - Konfigurationsmanagement [Westfechtel].
  - Das Nachfolge-Werkzeug FUJABA www.fujaba.de setzt diese Entwicklung fort
    - Findet direkte Anwendung in der modellgetriebenen Entwicklung.
  - Graphersetzungssysteme können einfach zum Bau von Interpretierern eingesetzt werden http://groove.cs.utwente.nl/

Konsequenz des Graph-Logik-Isomorphismus

- Wegen des Graph-Logik-Isomorphismus ist ein nahtloser Übergang zu Werkzeugen möglich, die mit Logik spezifiziert sind.
- Daher können in einer SEU logikbasierte Werkzeuge mit ersetzungsbasierten Werkzeugen eng zusammenarbeiten:
  - die unterliegenden Graphen der Artefakte können je nach Bedarf als Fakten oder Graphen interpretiert werden.

Ausblick

- Der Einsatz von Logik, Graphersetzungssystemen und Termersetzungswerkzeugen in Entwicklungsumgebungen ist noch kein Standard.
- Es ist aber vorauszusehen, dass diese generischen Werkzeuge wesentliche Bestandteile liefern werden,
  - sobald sie schnell und robust genug geworden sind
  - die Ausbildung die Konzepte in genügender Tiefe und Breite gelehrt hat.
- Viele der im Folgenden vorgestellten Werkzeuge können vereinheitlicht werden, indem Logik- und Ersetzungsspezifikationen Werkzeugt beschreiben und zu ihrer Generierung eingesetzt werden
- Damit könnten Werkzeugkomponenten und -rahmenwerke entstehen,
  - Die die Kosten für den Softwarewerkzeugbau senken
  - In einer SEU wären Logikmaschinen und Ersetzungssysteme Standardwerkzeuge