Software-Entwicklungswerkzeuge
Kap. 10 - Einführung

1

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de
WS 13/14-0.2, 14.10.13

1) Taxonomie von Werkzeugen
2) Werkzeug-Grundtypen
3) Werkzeuglandschaft
4) Graph-Logik-Isomorphismus

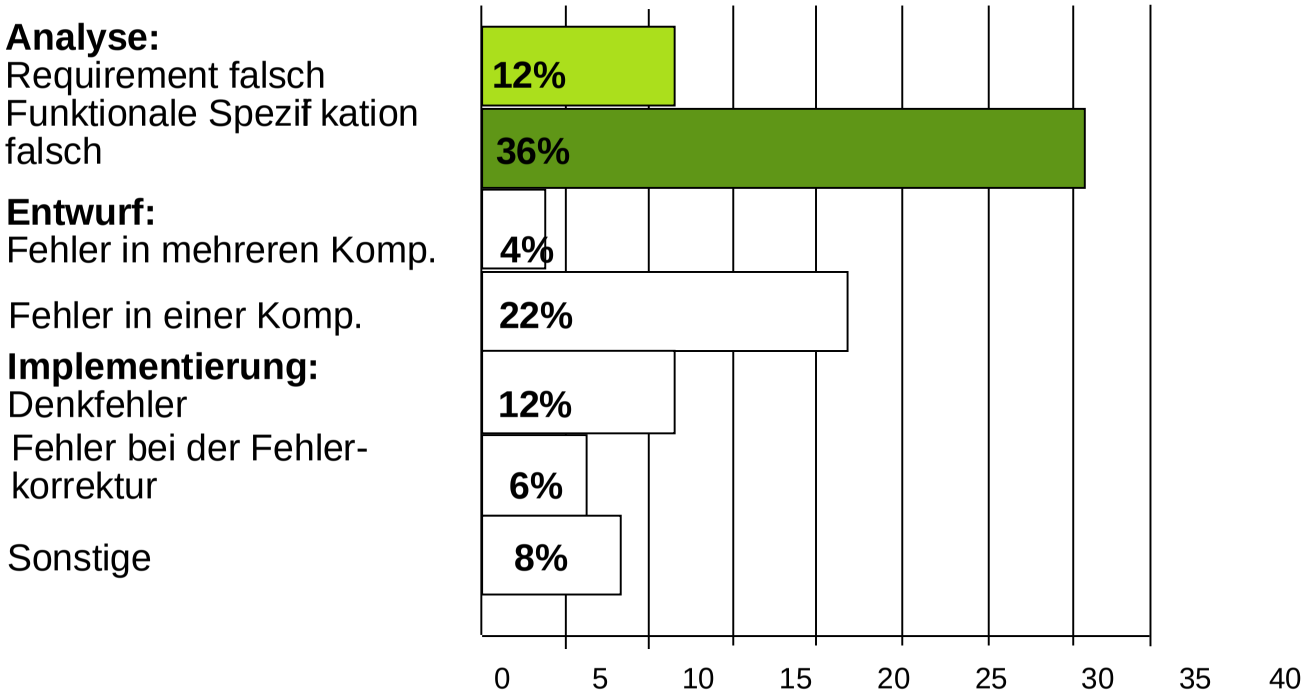
Software ist strategisch

2

- **Wertschöpfung** aus der Softwareentwicklung nach BMBF-Studie ca. 25,5 Mrd. EUR
  - bei Wachstumsrate von 12 % für 2003 etwa 38 Mrd. EUR
  - Bei Produkten der Telekommunikation und des Maschinen- und Anlagenbaus beträgt der Softwareanteil 75-80% der Herstellungskosten (steigend)
  - Komplexe Vermittlungsanlagen bis zu 6000 Mannjahre
  - Ein Mobiltelefon enthält ca. 250.000 lines of code (LOC)
- **Arbeitsplätze**:
  - Mehr als 65% der Berufstätigen arbeiten mit dem Computer, 95% der verkauften Rechner ging in Haushalte, mehr als 400 Mio. Server im Internet.
  - Aufwand zur Schaffung von Arbeitsplätzen gering, da zunächst Dienstleistungsgeschäft
- **Wachstum**: Die Zuwachsraten im Softwaremarkt liegen überdurchschnittlich hoch. Für
  - softwarebezogene Dienstleistungen 5,9%
  - Software 7,2%
  - davon Anwendungssoftware 8,8%
- **Kosten** der Softwareproduktion steigen ständig, weltweit > $ 250 Billionen im Jahr
  - Wartungskosten betragen etwa 60% der Softwarekosten
  - Softwaresysteme sind hochgradig heterogen, oft Software-Landschaften, die in mehreren Technikräumen konstruiert werden (XML, Java, C, C++, Simulink, etc.)
- **Aber**: Nur ca. 30% der Unternehmen nutzen moderne Methoden und Werkzeuge, um ihre Kosten zu reduzieren

Fehlerquellen bei der Software-Entwicklung

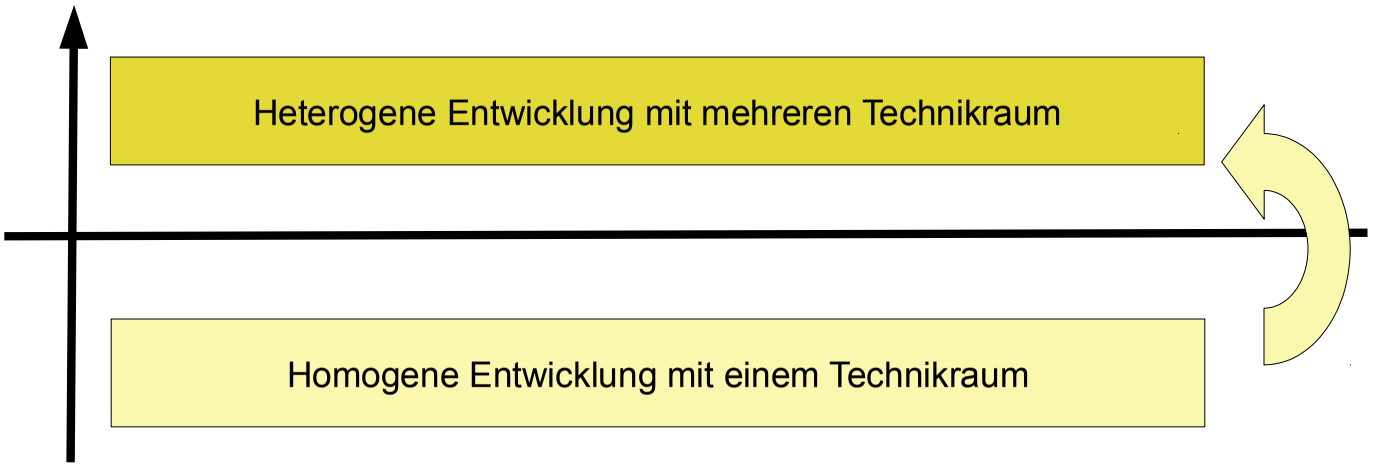
- Wichtig ist daher der Einsatz von Werkzeugen in frühen Phasen



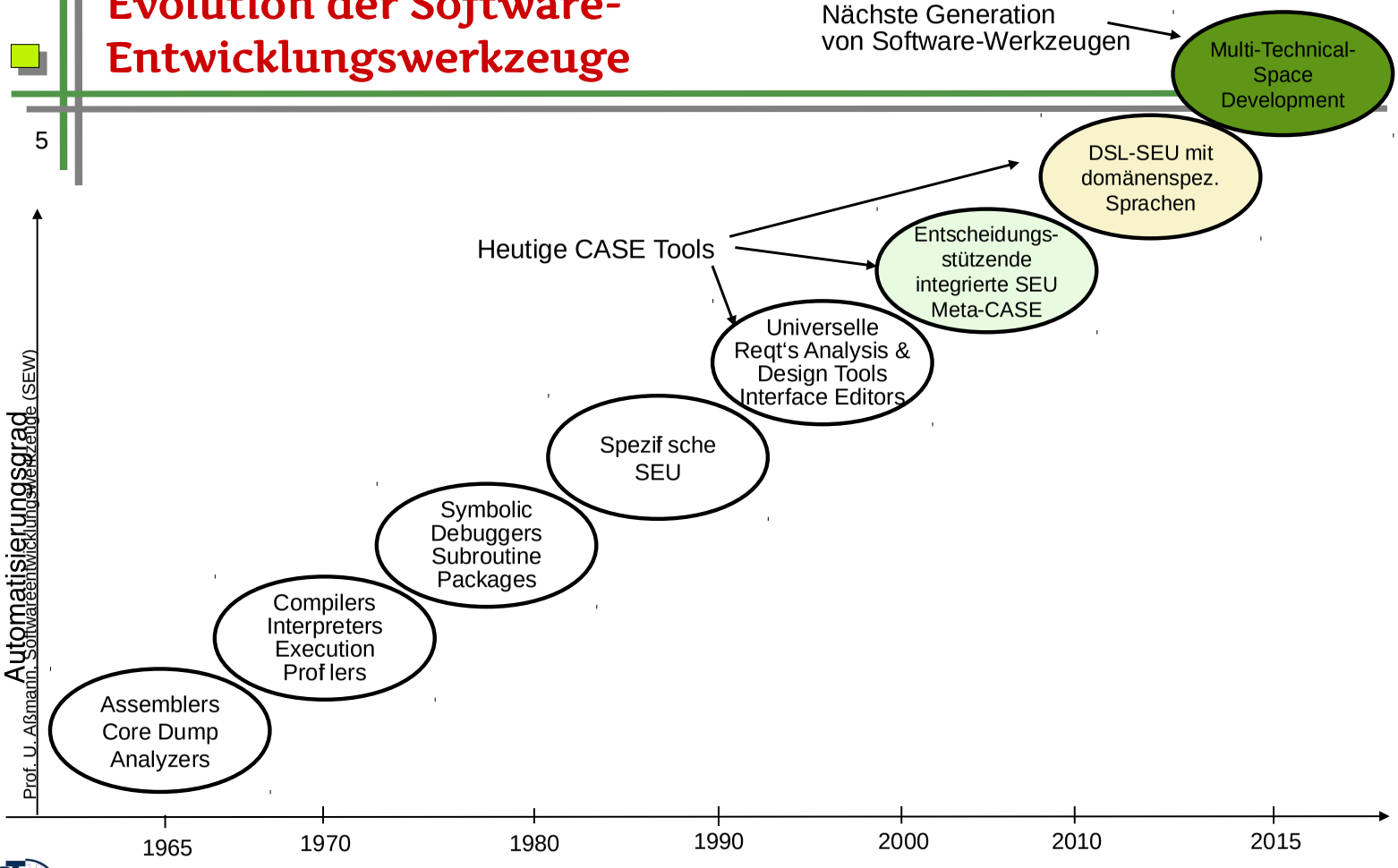
Quelle: Balzert, H.(Hrsg.): CASE - Auswahl, Einführung, Erfahrungen; BI-Wissenschaftsverlag 1993, S.59

Reifestufen von Softwarefirmen

- Viele Firmen arbeiten nur auf dem Niveau von *homogener Softwareentwicklung* in *einem* Technikraum
- Um auf höhere Stufe zu gelangen, *heterogene Softwareentwicklung,* die für komplexe Softwaresysteme nötig ist, müssen Werkzeuge eingesetzt werden



Evolution der Software-Entwicklungswerkzeuge



Quelle: nach [Fisher91, S.20]

10.1 Werkzeuge und Software-Entwicklungs-umgebungen (SEU)

10.0.1 Begriffs-Definitionen

Warum will der Mensch Werkzeuge einsetzen?

Ein **Werkzeug** ist ein Hilfsmittel, um Dinge schneller, präziser zu erledigen als von Hand.
Ein **IT-Werkzeug** ist ein Werkzeug, das im Rechner läuft und Informationen verarbeitet.
Ein **Software-Werkzeug** ist ein IT-Werkzeug, das Software bearbeitet.
Eine **Werkzeugmaschine** ist ein Werkzeug, mit dem man ein anderes Werkzeug herstellt.
Eine **Software-Werkzeugmaschine** ist ein Werkzeug, mit dem man andere Software-Werkzeuge herstellt.

- Werkzeuge werden eingesetzt
  - Zur Automatisierung
  - Zur Vereinfachung
- Extensive Werkzeugnutzung zeichnet den Menschen gg. allen anderen Lebewesen aus
- SW-Werkzeuge können zum Bau von Werkzeugen eingesetzt werden
- SW-Werkzeugmaschinen sind die Grundlage aller Produktivität
- SW-Werkzeugmaschinen sind die Grundlage des Wohlstands

"Tools and Material"-Metapher (TAM)

Tool:

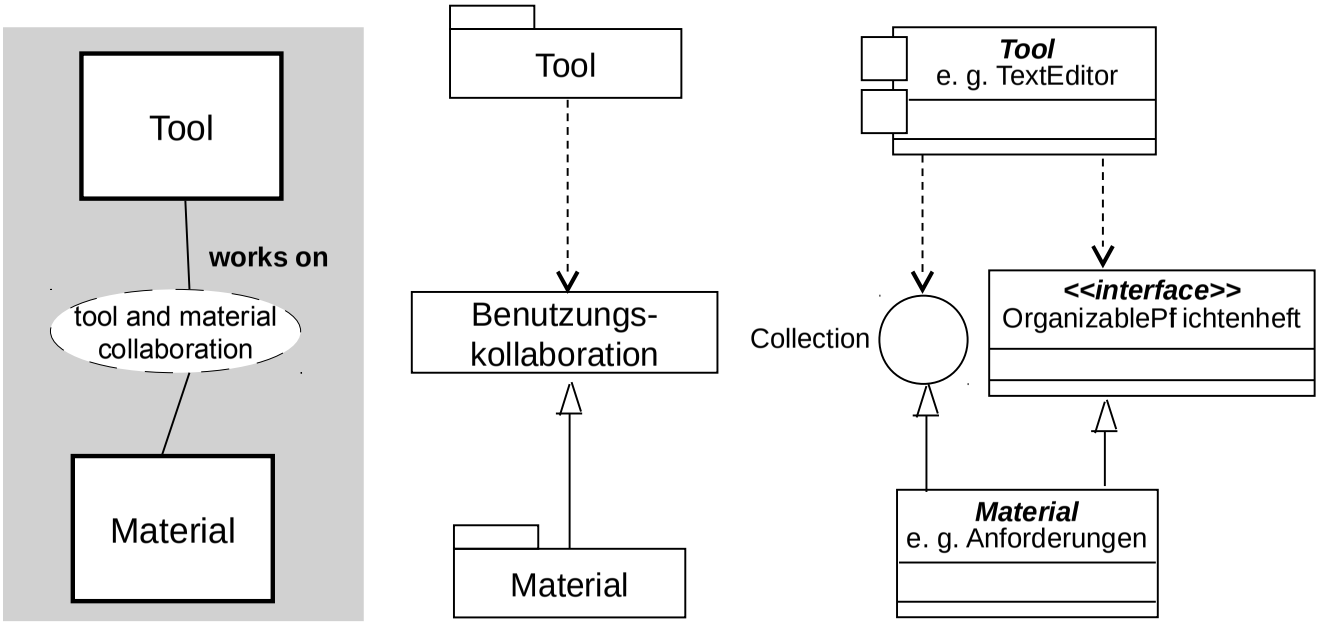
- ist ein aktives Objekt, das Menschen benutzen können zum Umgestalten oder zum Verändern von **Material**, um eine spezifische Aufgaben zu lösen.
- **Tools** sind normalerweise geeignet für unterschiedliche Aufgabenbereiche, um verschiedenes Material zu bearbeiten.
- Viele konzeptuelle Eigenschaften der **Tools** können auf Software-Entwicklungswerkzeuge übertragen werden. Sie sollten für unterschiedliche Aufgaben und verschiedenes Material innerhalb von Softwaresystemen geeignet sein.

Material:

- ist ein passives Objekt, das Teil eines Arbeitsresultates wird. **Material** wird unter Benutzung von **Tools** verändert nach einem domänenspezifischen Konzept.
- Das Zusammenspiel von Tools und Material wird durch eine **Kollaboration (Rollenmodell)** ausgedrückt (siehe Kurse Softwaretechnologie, DPF).

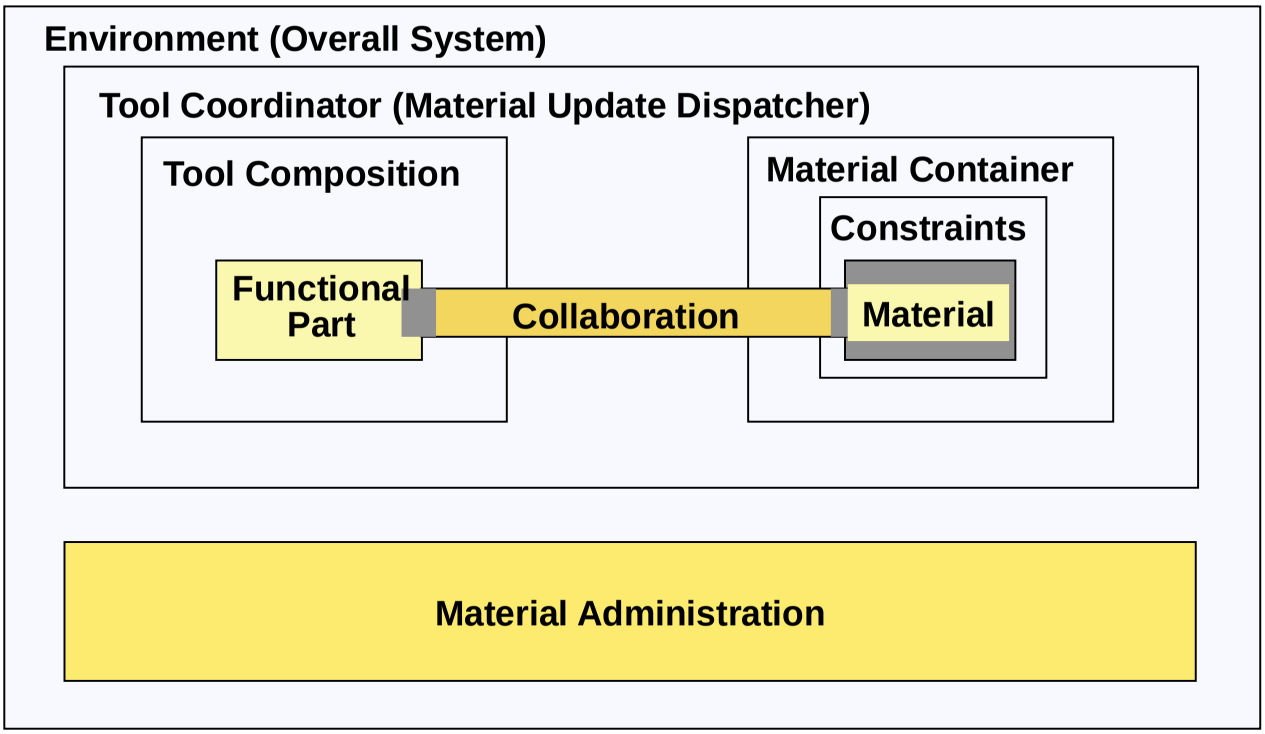
[Züllighoven, Heinz: Object-Oriented Construction Handbook; dpunkt.verlag 2005]

Tool and Material - Kollaboration



Quelle: Züllighoven, H.: Object-Oriented Construction Handbook; dpunkt.verlag Heidelberg 2005, S. 87

TAM Patterns for Tool Integration



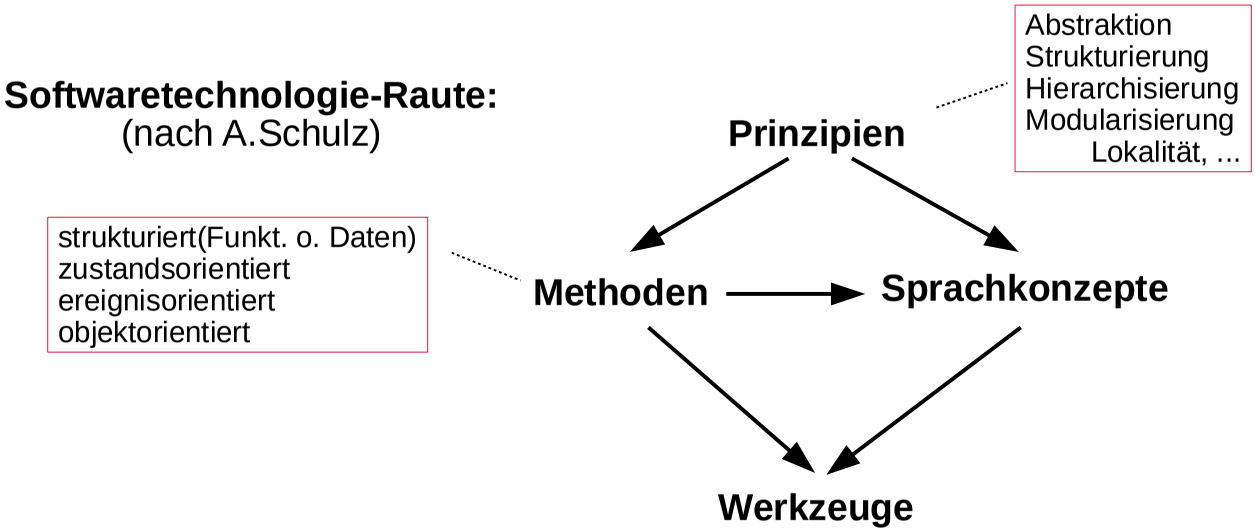
Quelle: Riehle, D., Züllighoven, H.: Pattern Languages of Program Design; Reading, Massachusetts: Addison Wesley 1995, Chapter 2, S. 9-42

Siehe auch Kapitel „Repository“

Definitionen

Software-Werkzeuge sind Programme (Software, Hilfsmittel), die Vorgehensweisen, *Prinzipien, Methoden* und *Sprachkonzepte* rechnergestützt umsetzen und den Benutzer bei der Software-Entwicklung unterstützen (nach [6, S.204]).

Softwaretechnologie-Raute:
(nach A.Schulz)



Software-Entwicklungsumgebungen (SEU)

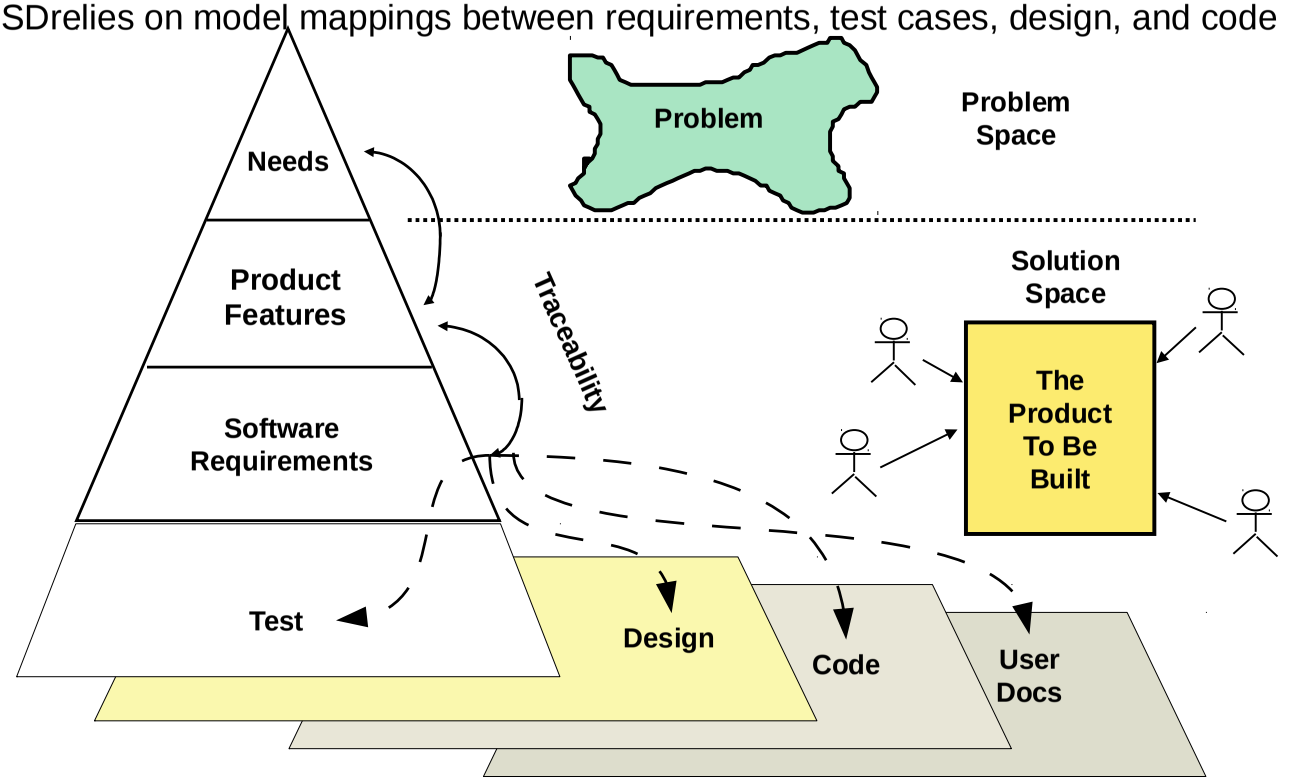
Eine **Software-Entwicklungsumgebung (SEU, integrated development environment, IDE)** besteht aus einer **strukturierten Menge integrierter Werkzeuge und Bausteine,** die ein Team bei allen in der Software-Entwicklung anfallenden Tätigkeiten unterstützen soll einschließlich einer einheitlichen Methodik für seine Nutzung.

- Eine SEU ist also eine komplexe Software-Werkzeugmaschine
  - **Computer aided Software Engineering (CASE)**, CASE-Umgebung, CASE Environment
- Eine **Meta-CASE** ist eine SEU, in der viele verschiedene Sprachen behandelt werden können und die *heterogene Softwareentwicklung* unterstützt
- Ältere Bezeichnungen in der Literatur:
  - Integrated Computer Aided Software Engineering (I-CASE)
  - Software-Produktionsumgebung (SPU), Integrated Software Factory (ISF)
  - Software Engineering Environment System (SEES)
  - Integrated Project Support Environment (IPSE)
  - Integrated Software Engineering Environment (ISEE)

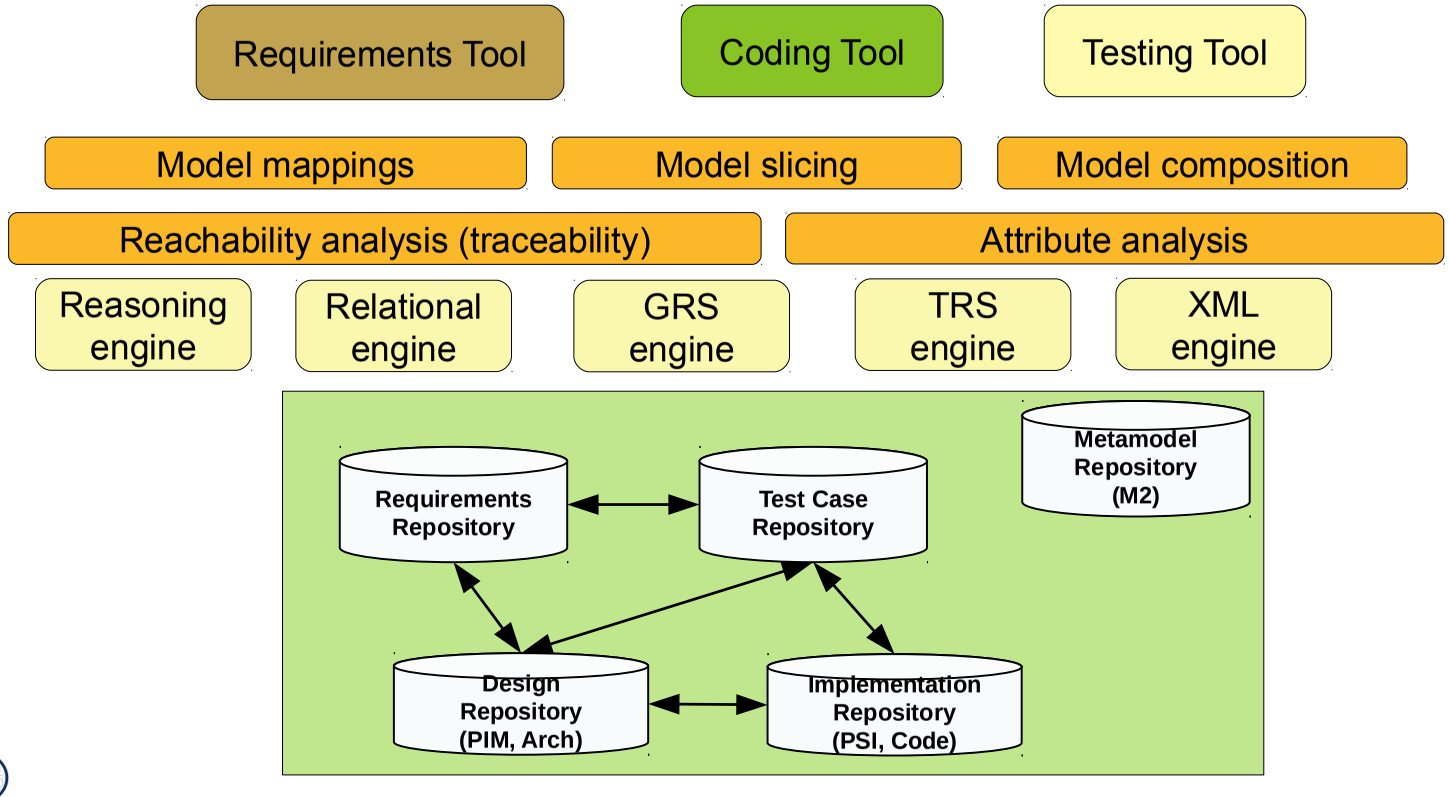
Nagl. M.: Software-Entwicklungsumgebungen: Einordnung und zukünftige Entwicklungslinien; Informatik-Spektrum 16(1993) H.5, S. 273-280

Model-Driven Software Development

- MDSD systematically connects the customer's problems, the system's requirements, testing, design, coding, and documentation and develops these models in coordination
- MDSDrelies on model mappings between requirements, test cases, design, and code



Tools in an Integrated Development Environment (IDE, SEU) for MDSD

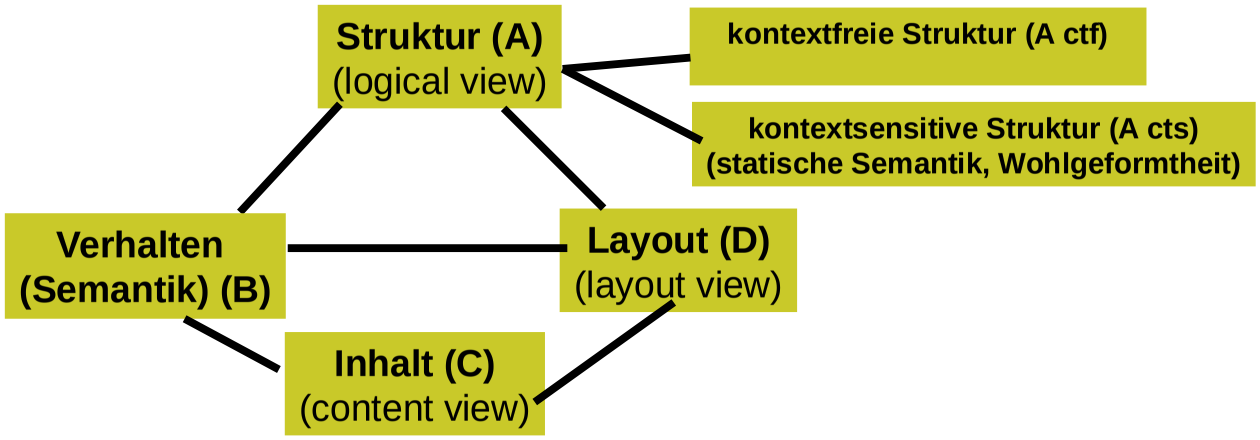


Method Engineering (Process Engineering)

Process Engineering (Method Engineering) is the discipline of constructing and running processes for a team of people to conduct a project.
Software Process Engineering (Software Method Engineering) focuses on software development processes.

10.1.2 Aufbau und prinzipielle Funktion von Software-Entwicklungswerkzeugen

Aspekte von Artefakten (Dokumente, Modelle, Code)

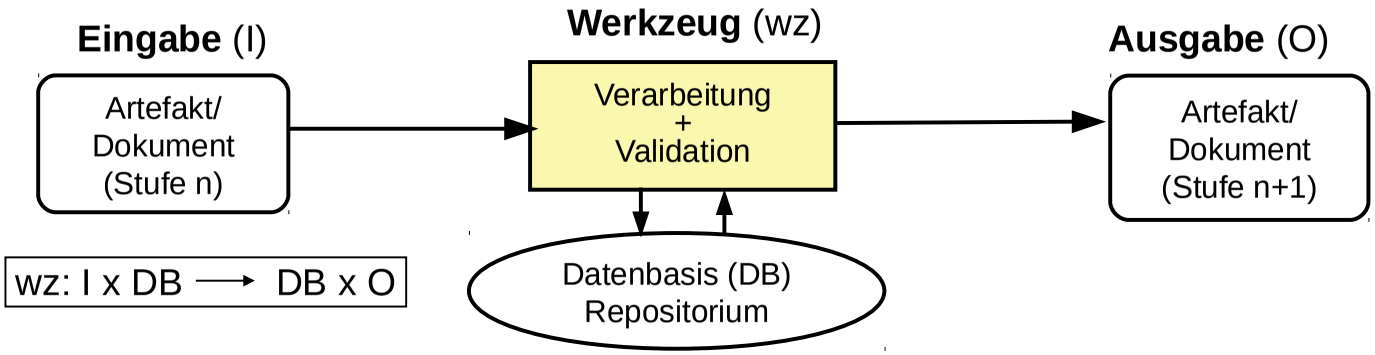


- **Struktur:** log. Einheiten, wie Gliederung, Überschriften, Fußnoten, Köpfe, Verweise
  - **kontextfreie Struktur**
  - **kontextsensitive Struktur mit Konsistenzbedingungen (statische Semantik)**
- **Semantik:** Programme besitzen eine *Bedeutung (dynamische Semantik, Verhalten)*
- **Inhalt:** Text, Graf ken, Bilder, Bitmuster, elektron. Erscheinungsformen
- **Layout:** Ausgabeanordnungen und -vorschriften für log. und inhaltliche Elemente

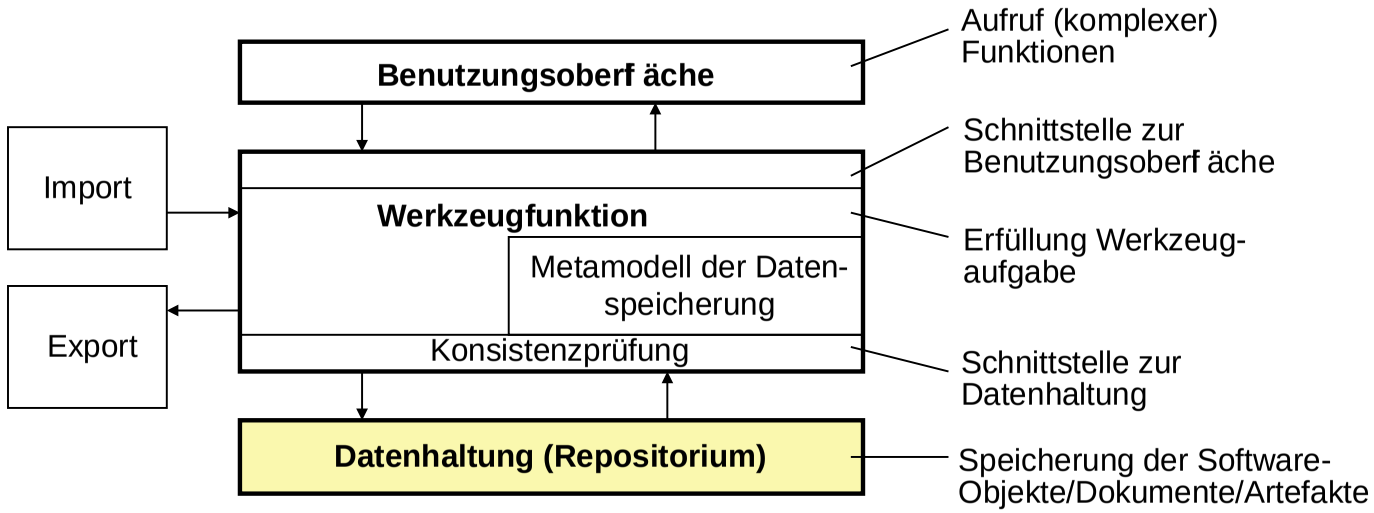
Werkzeugfunktionen

- **Codezentrierte Werkzeuge:** Ursprünglich wurden nur einzelne grundlegende Komponenten der Software-Entwicklung wie Compiler, Editoren oder Testhilfen als Werkzeuge bezeichnet
  - **Programme** (Code, Codeschablonen/Templates, Fragmente, Snippets)
- **Modellzentrierte Werkzeuge:** Im Laufe der Zeit kamen viele spezialisierte Entwicklungs- und Administra-tionswerkzeuge hinzu:
  - Herstellung und Verarbeitung von **Artefakten (Arbeitsergebnissen)** in einer textuellen, grafischen oder mathematischen Notation
    - Freitext (Prosa, Bilder, formatierte Texte)
    - Dokumente
    - Modelle (Beschreibungen) und Spezifikationen (Vorschriften)
    - Programme
  - **Konsistenzprüfung** auf Wohlgeformtheit von einzelnen Artefakten und zusammengehörigen Artefaktbeständen (**statische Semantik**)
    - Ausführen der **dynamischen Semantik**
    - --> Werkzeugprüfung/-Unterstützung (2)
  - Unterstützung des Prozess-Engineering (method engineering)
    - Unterstützung von **Methoden** und einzelner Entwicklungsschritte (Entwurf, Testen,...)
    - Unterstützung von **Phasen- und Vorgehensmodellen**
      - --> Vorgehensweise/Methodik (3)

Werkzeug - Wirkungsschema



Werkzeug – Grobarchitektur, logische Sicht



Quelle: [3 Bal-II, S. 604]

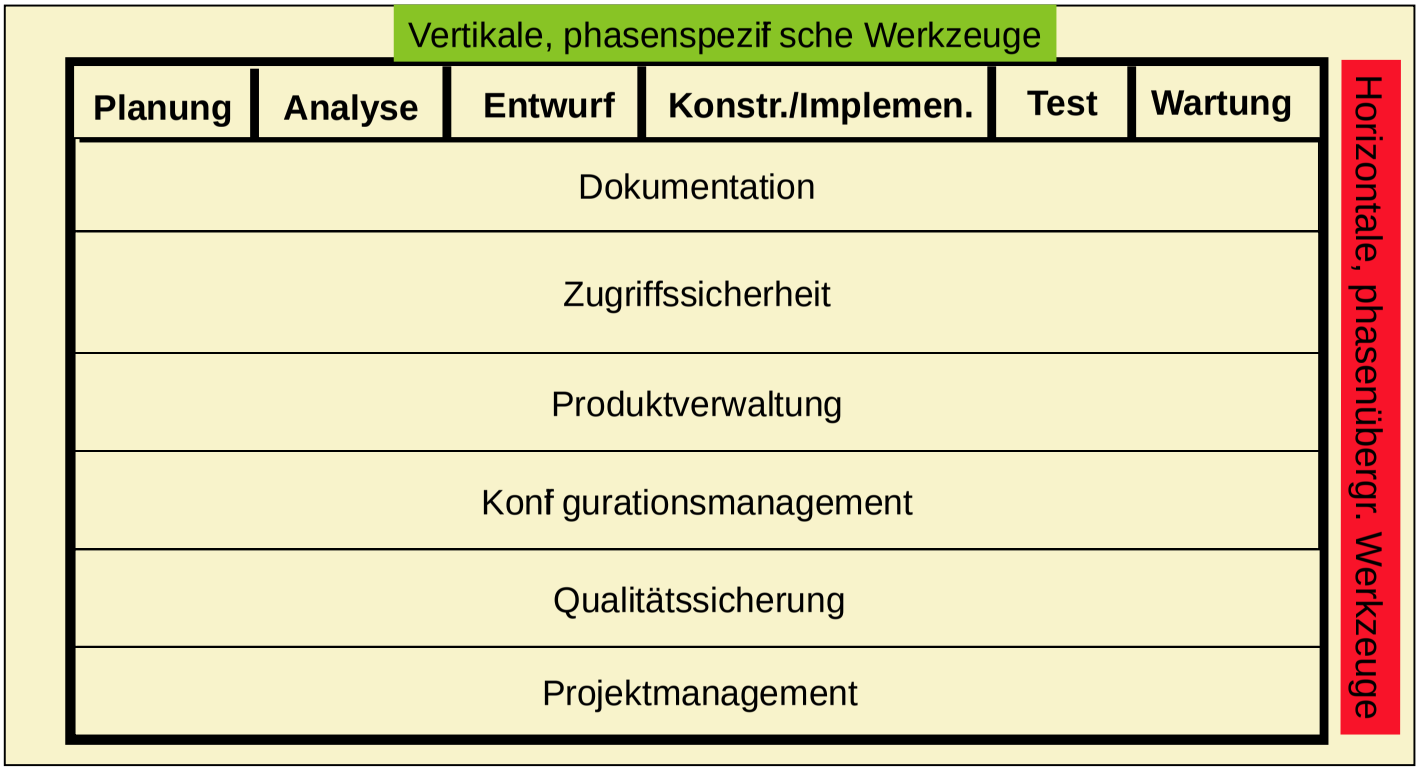
Dokumenttypen (Artefakte) der Softwareentwicklung

- Freitext
  - z.B. Anforderungsspezifikation, Entwurfsspezifikation, Programmbeschreibungen,...
- Modelle in Form von Diagrammen (Graphen,)
  - z.B. Analyse- und Entwurfsspezifikation (UML-Diagramme), Programmstrukturen,...
  - Ontologien
- Grafiken
  - komplexe visuelle Darstellungen in 2-D oder 3-D, Visualisierungen
- Bäume
  - S-Expressions (Lisp, Scheme)
  - XML-Bäume
  - Xcerpt-Bäume, JSON-Bäume
- Tabellen
  - z.B. Relationen, Testfalltabellen
- Code
  - z.B. Pseudocode, Codegerüste, Quellcode

Quelle: Denert, E.: Dokumentenorientierte Software-Entwicklung; Informatik-Spektrum 16(1993) H. 3, S. 159 - 164

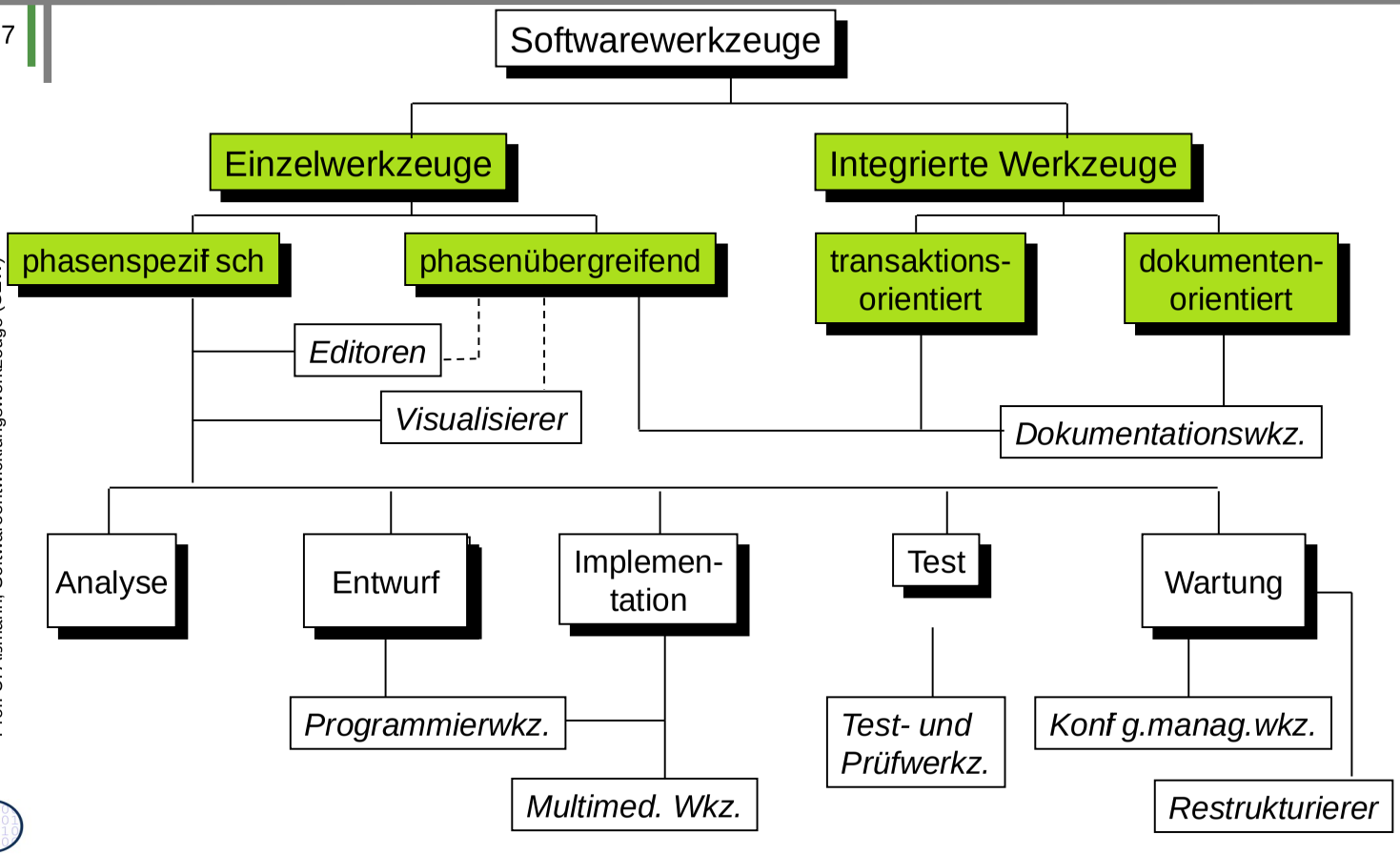
10.2 Werkzeuggrundtypen

Entwicklungsaufgaben und Werkzeuge

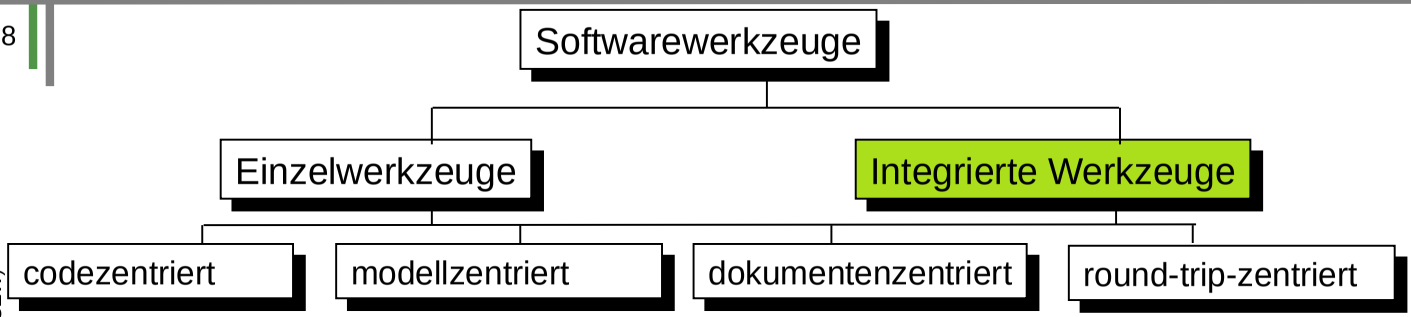


Quelle: Forte, G., McCulley, K.(Hrsg.): CASE Outlook: Guide to Products and Services; CASE Consulting Group, Lake Oswego, Orl., 1991

Eine Grobgliederung von Software-Entwicklungswerkzeugen

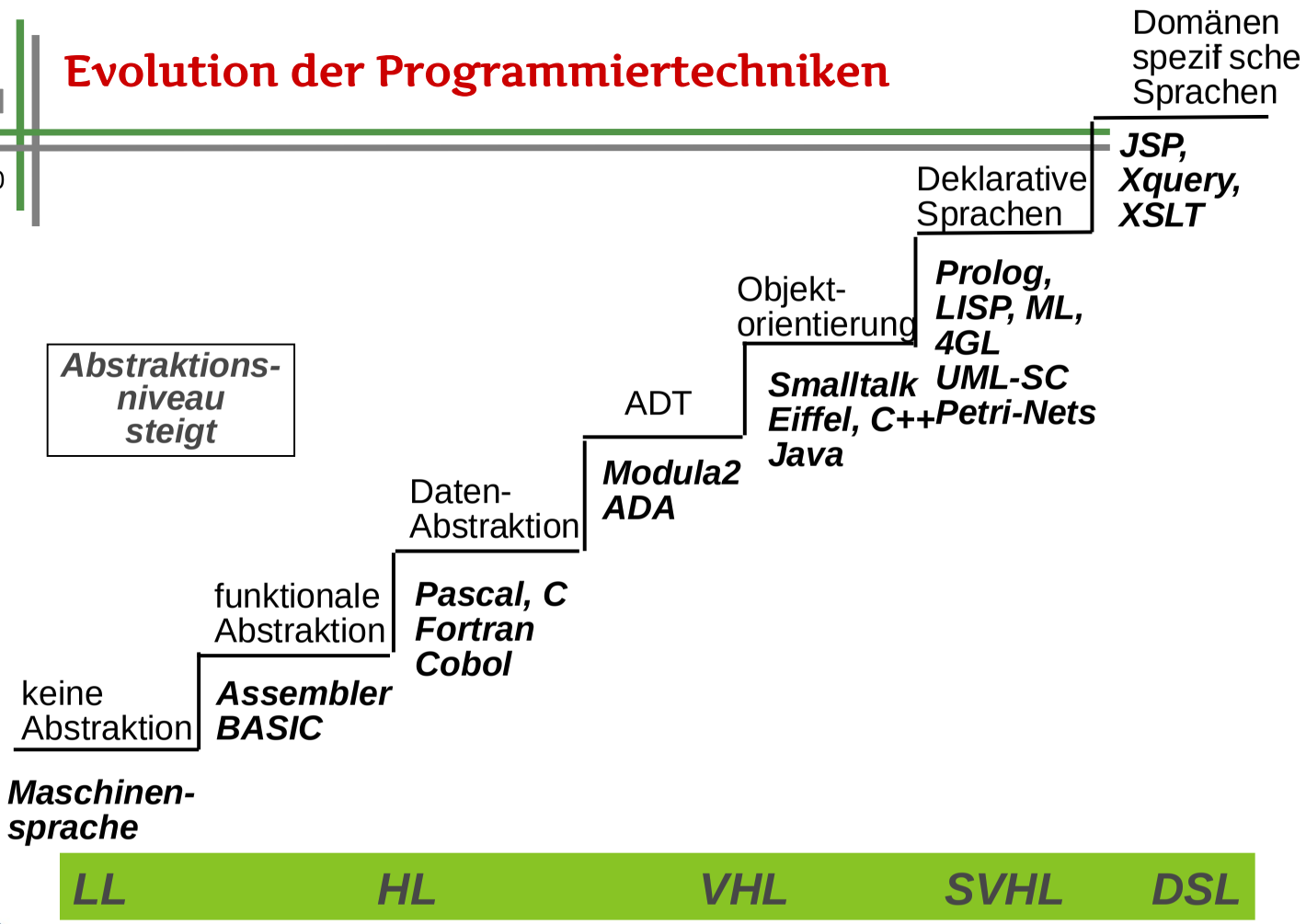


Eine Grobgliederung von Software-Entwicklungswerkzeugen



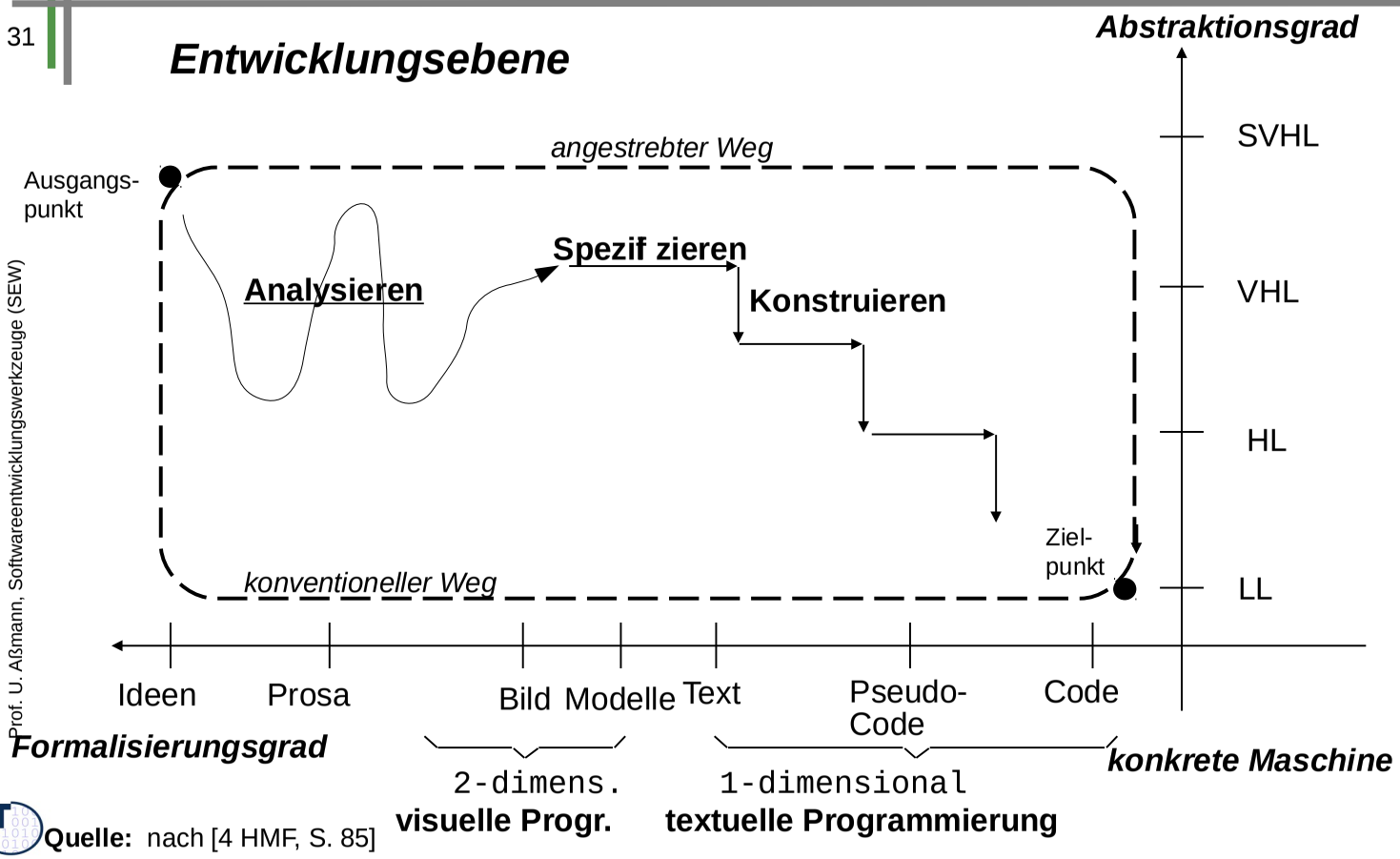
10.3 Werkzeug-Landschaft nach Hesse

Evolution der Programmiertechniken



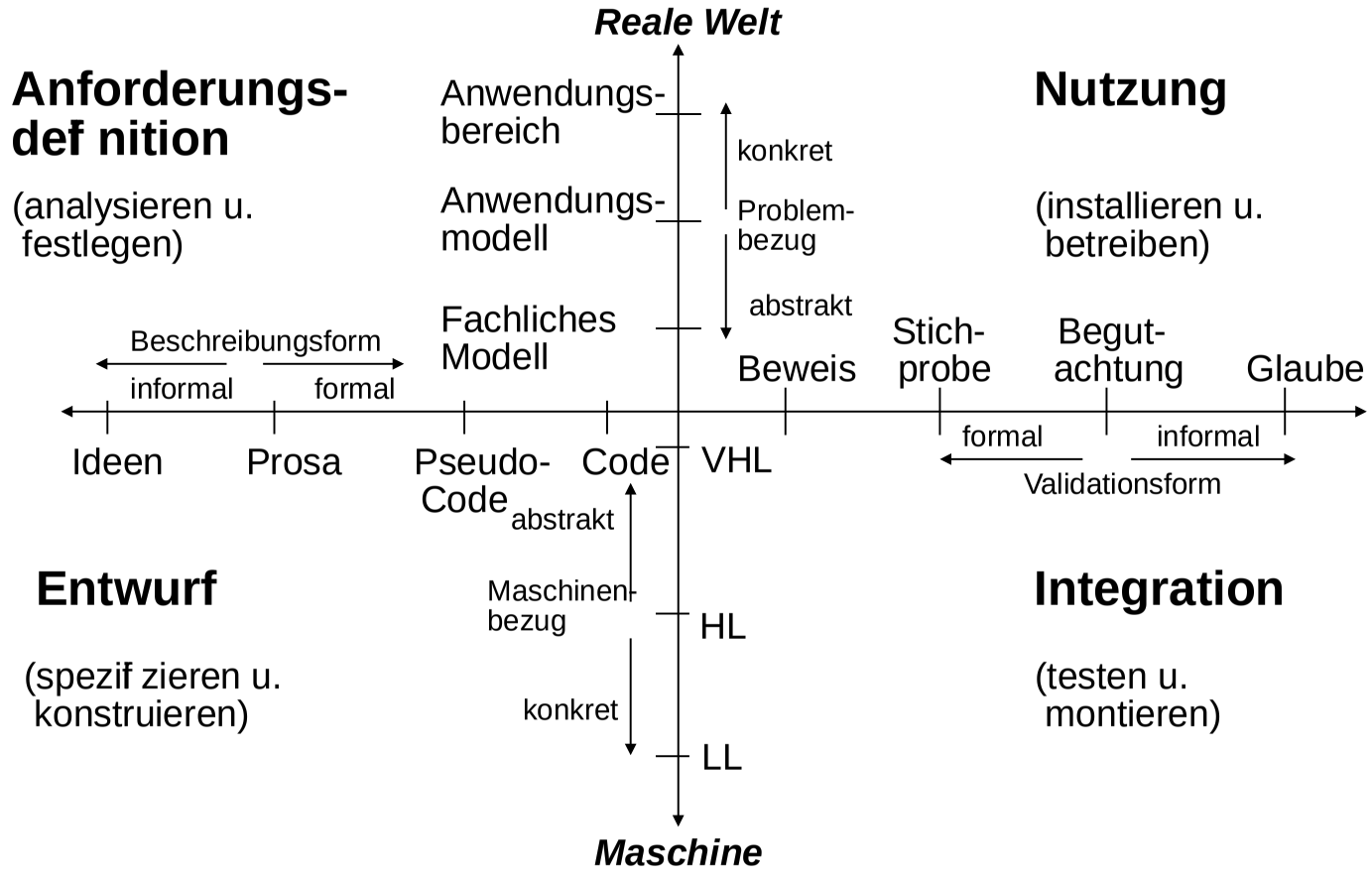
Quelle: nach [1, S. 98]

Abstraktion der Softwareentwicklung von Hesse



Quelle: nach [4 HMF, S. 85]

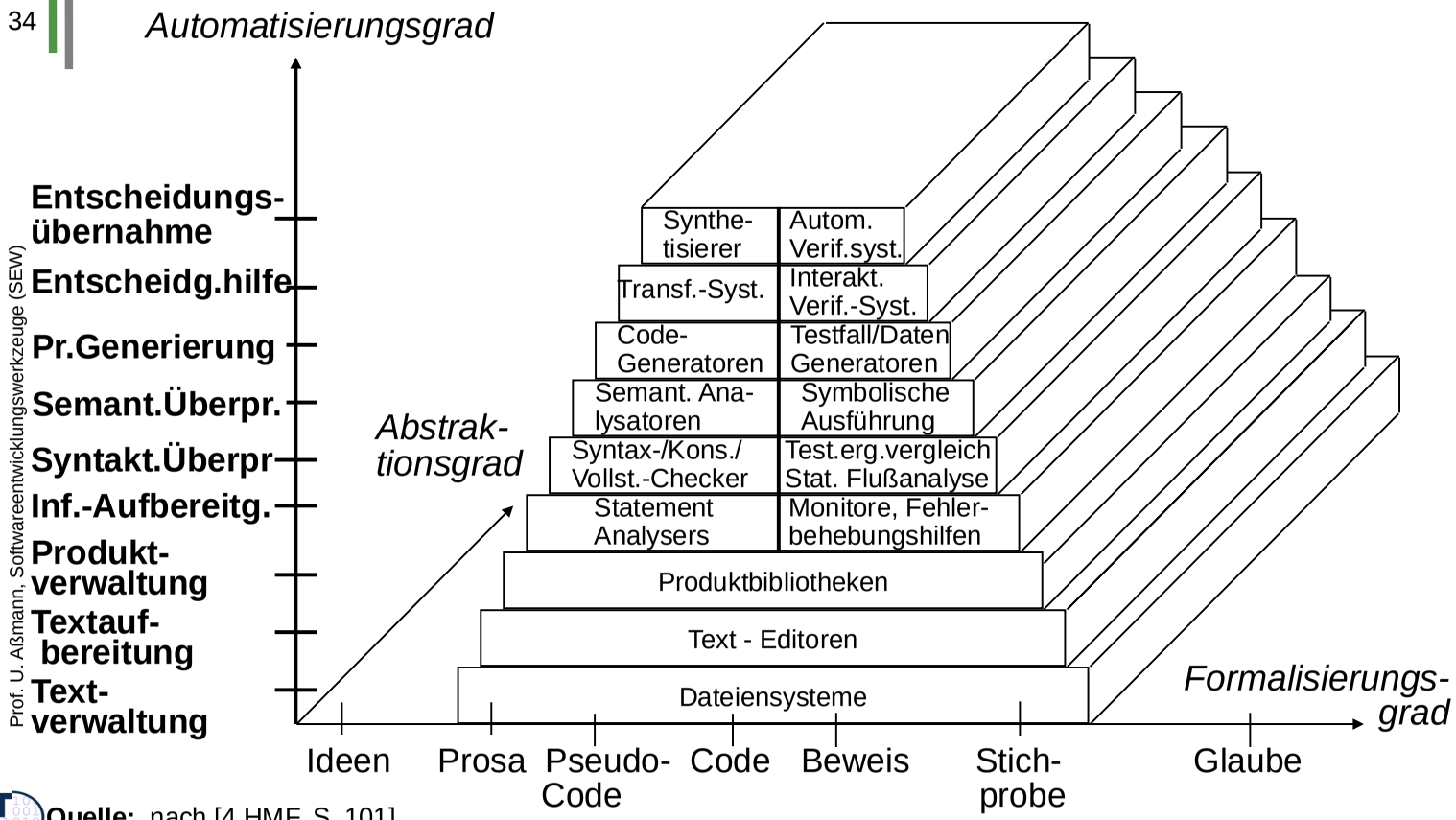
Software-Entwicklungsquadranten von Hesse



Automatisierungsgrad von Werkzeugen von Hesse

Nr.	Stufe	Funktion
9	Entscheidungs-übernahme	Automatisierung der Übergänge zwischen Entwicklungsschritten durch kooperierende, inferenzbasierte Werkzeuge [20]
8	Entscheidungs-Hilfe	Interaktive Transformationssysteme z.B. bei der Restrukturierung sowie bei der interaktiven Verif kation
7	(Produkt-)Gene-rierung	Automatische Erzeugung von Codegerüsten (Programmen) aus Ent-würfen und Testfällen/Testdaten aus der Anforderungsspezif kation
6	Semantische Überprüfung	Analyse z.B. des kontext-sensitiven Teils formaler Spezif kationen und andere die Programmausführung betreffende Inhalte
5	Syntaktische Überprüfung	Vollständige synt. Überprüfung formaler Spezif kationen durch „Syntax-Checker“, Parser, Flussanalysen usw.
4	Informations-Aufbereitung	Syntaktische Analyse von bestimmten formal-sprachlichen Informa-tionen, Ausgabe von Inkonsistenzen, Fehlern, Querbezügen
3	Produktverwal-tung	Manipulieren und Verwalten von wohldef nierten „Teilprodukten“, Sicherung der konsistenten Verwahrung von Versionen
2	Textaufbereitung	Fortgeschrittene Editorfunktionen, wie abschnittsweises Kopieren, Copy, Cut, Paste, Layoutfunktionen, Suchen + Ersetzen,...
1	Textverwaltung	Eingabe, Speicherung, Ausgabe von Texten mit Hilfe eines Dateisystems (normale Werkzeugfunktion)

Softwaretechnologie – Landschaft von Hesse



Prof. U. Aßmann, Softwareentwicklungswerkzeuge (SEW)

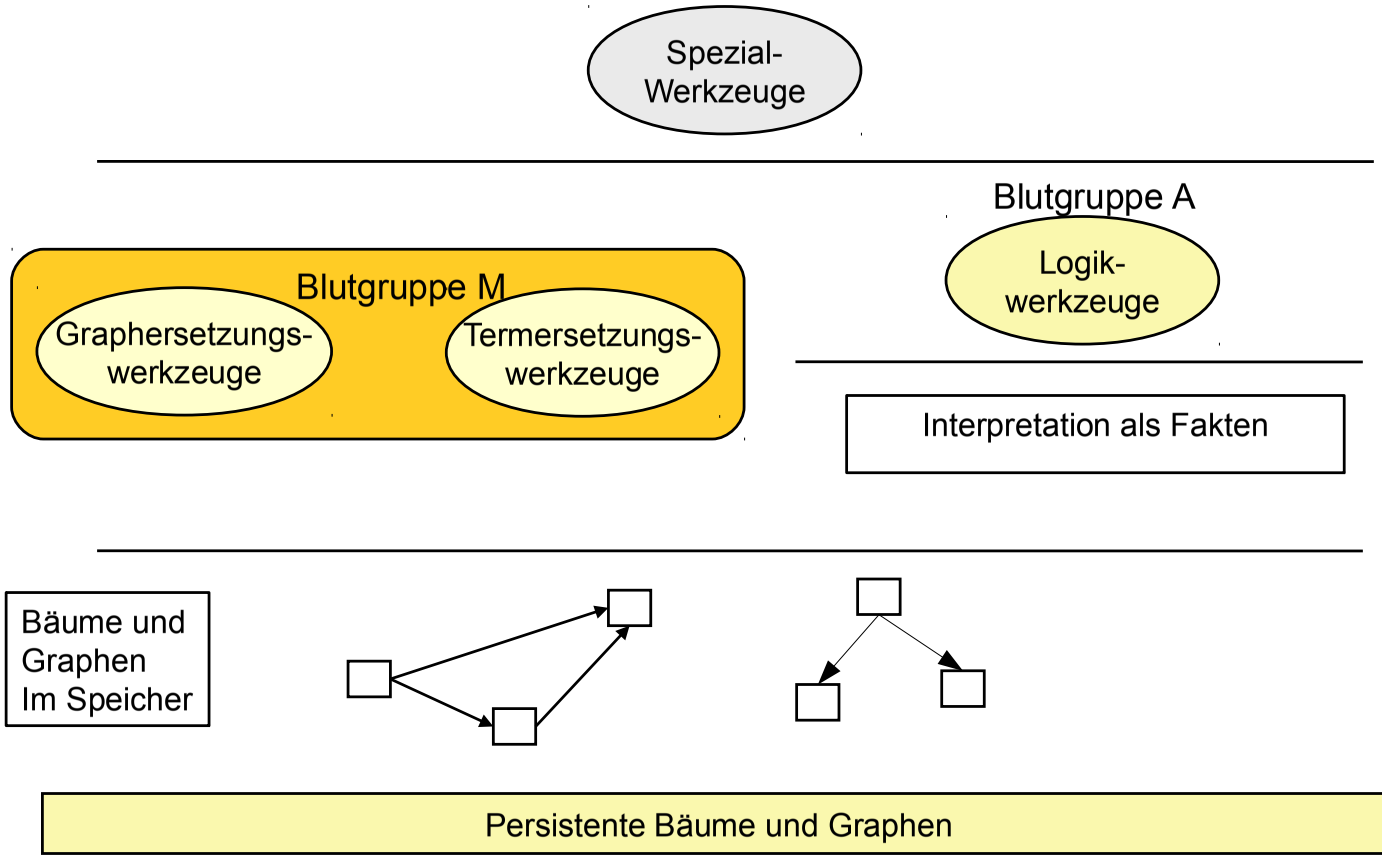
Quelle: nach [4 HMF, S. 101]

10.4 Der Graph-Logik-Isomorphismus

Der Graph-Logik-Isomorphismus

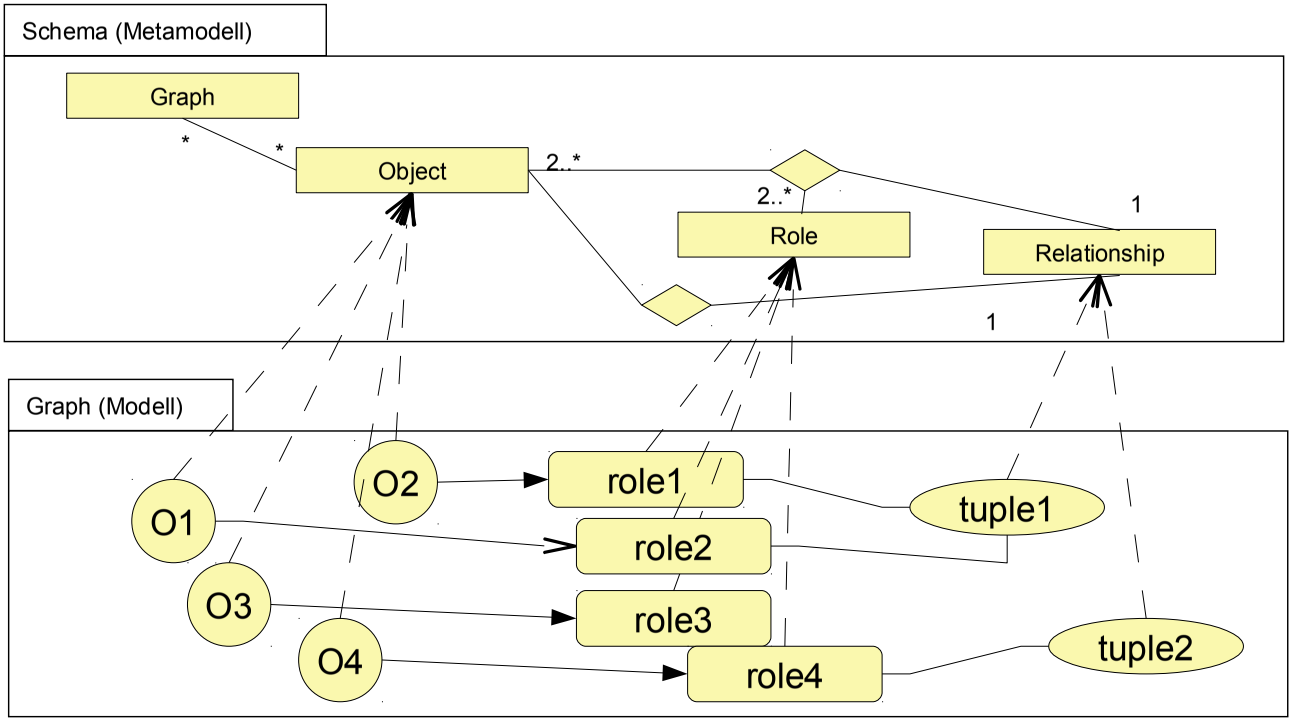
- Jeder Graph kann als Faktenbasis einer Logikmaschine abgelegt werden.
- Jede Faktenbasis kann als Graph interpretiert werden
  - binär: Graph
  - n-är: Hypergraph
- Logikmaschinen und Graphtransformations-Werkzeuge können zu guten Teilen ausgetauscht werden
- Die *Metamodellierung* setzt auf beiden Ansätzen zugleich auf

SEU mit Ersetzungs- und Logik-Werkzeugen



Typisierte Graphen (Modelle und Metamodelle)

- Graphen können typisiert sein, aber die Schemata können unterschiedlich aussehen (→ Metamodellierung)



Anhänge