

# Software-Entwicklungswerkzeuge

## Kap. 10 - Einführung

Prof. Dr. Uwe Aßmann  
 Technische Universität Dresden  
 Institut für Software- und  
 Multimediatechnik  
 http://st.inf.tu-dresden.de  
 WS 11/12-0.3, 08.10.11

- 1) Taxonomie von Werkzeugen
- 2) Werkzeug-Grundtypen
- 3) Werkzeuglandschaft
- 4) Einführung in die Effektkategorien
- 5) Graph-Logik-Isomorphismus

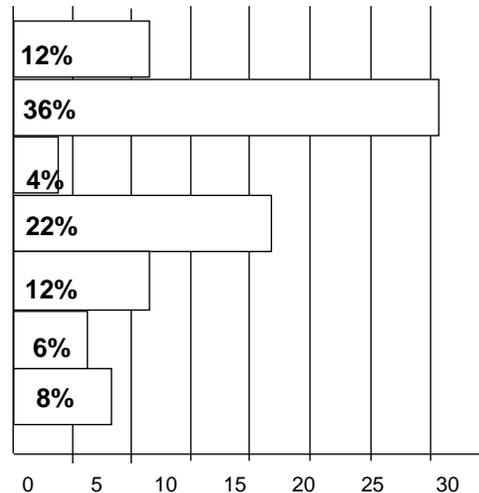
## Notwendigkeit des Einsatzes von Software-Entwicklungswerkzeugen

- ▶ **Kosten** der Softwareproduktion steigen ständig, weltweit > \$ 250 Billionen im Jahr
- ▶ 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)
- ▶ Mehr als 65% der **Berufstätigen** arbeiten mit dem Computer, 95% der verkauften Rechner ging in Haushalte, mehr als 400 Mio. Server im Internet.
- ▶ Die **Zuwachsraten im Softwaremarkt** liegen überdurchschnittlich hoch. Für
  - softwarebezogene Dienstleistungen 5,9%
  - Software 7,2%
  - davon Anwendungssoftware 8,8%
- ▶ **Wartungskosten** betragen etwa **60%** der Softwarekosten
- ▶ Aber: Nur ca. 30% der Unternehmen nutzen moderne Methoden und Werkzeuge.

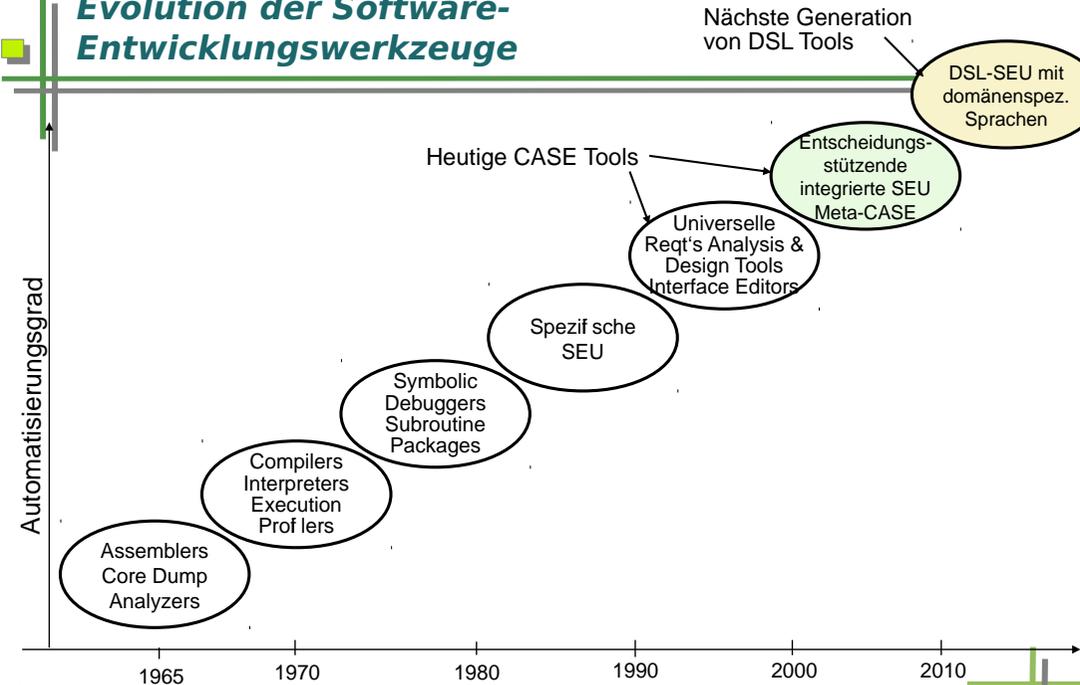
## Fehlerquellen bei der Software-Entwicklung

- ▶ Wichtig ist daher der Einsatz von Werkzeugen in frühen Phasen

**Analyse:**  
 Requirement falsch  
 Funktionale Spezifikation falsch  
**Entwurf:**  
 Fehler in mehreren Komp.  
 Fehler in einer Komp.  
**Implementierung:**  
 Denkfehler  
 Fehler bei der Fehlerkorrektur  
 Sonstige



## Evolution der Software-Entwicklungswerkzeuge



# 10.1 Taxonomie von Werkzeugen 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
- ▶ Werkzeuge können zum Bau von Werkzeugen eingesetzt werden
- ▶ Werkzeugmaschinen sind die Grundlage aller Produktivität
- ▶ Werkzeugmaschinen

## “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.

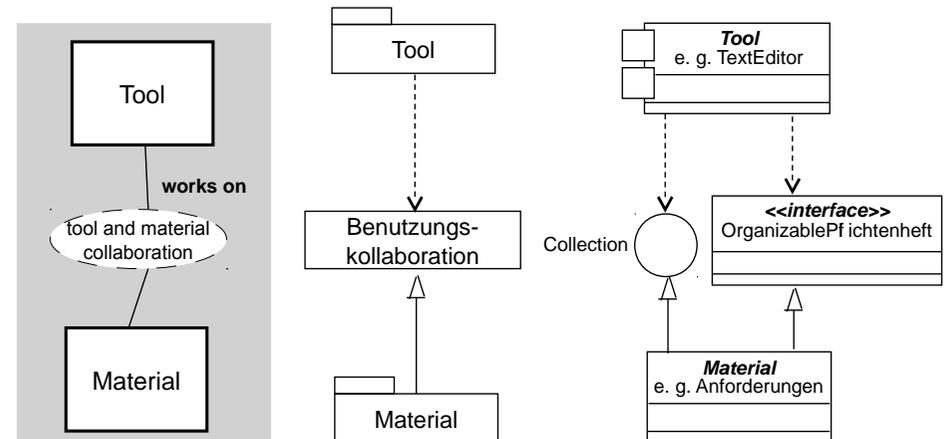
### Material:

- ist ein passives Objekt, das Teil eines Arbeitsergebnisses 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 „Softwaretechnologie“).

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

Diese Definitionen sind Basis einer allgemeinen Pattern Language für werkzeuggestützte Softwaretechnologie.  
 Siehe auch Kurs „Design Patterns and Frameworks“

## Tool and Material - Kollaboration

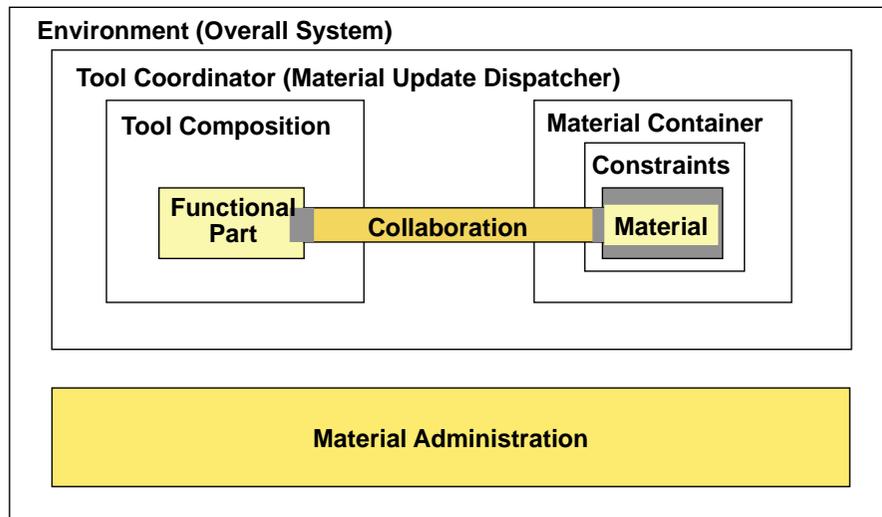


Conceptual Pattern

Design Pattern

Construction part

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



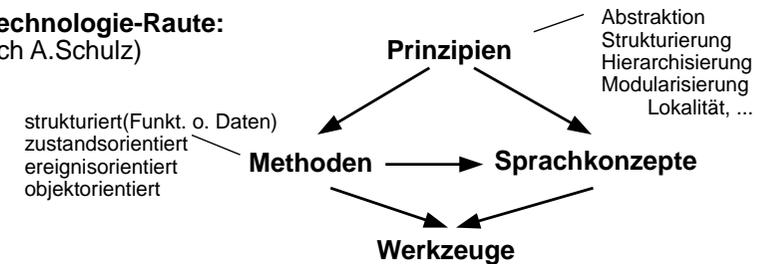
Quelle: Riehle, D., Züllighoven, H.: Pattern Languages of Program Design; Reading, Massachusetts: Addison Wesley 1995, Chapter 2, S. 9-42

Siehe auch Kapitel „Repository“

Prof. U. Aßmann, SEW 9

**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)



Prof. U. Aßmann, SEW 10

## Software-Entwicklungsumgebungen (SEU)

Eine **Software-Entwicklungsumgebung (SEU)** 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
    - Integrated Computer Aided Software Engineering (I-CASE)
  - Software-Produktionsumgebung (SPU)
  - Software Engineering Environment System (SEES)
  - Integrated Project Support Environment (IPSE)
  - Integrated Software Engineering Environment (ISEE)
  - Integrated Software Factory (ISF)

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

Prof. U. Aßmann, SEW 11

## Spannbreite des Begriffes SEU

umfasst:

- (1) eine auf einen Anwendungsbereich abgestimmte **Modellierungs- oder Arbeitsumgebung**, in der der Anwender direkt seine Gedankenwelt vorfindet und nicht mehr im klassischen Sinne programmiert;
- (2) eine **Auswahl von Werkzeugen** und Bausteinen für einen Anwendungsbereich, die dort angewandte Methoden und Programmiersprachen unterstützen;
- (3) eine **Sammlung** vorgegebener, mehr oder minder brauchbarer **Programmbausteine** für einen Anwendungsbereich;
- (4) eine abgestimmte Softwaretechnik-Arbeitsumgebung zur **Erstellung beliebiger Softwaresysteme** auf eine oder mehrere Programmiersprachen abgestimmt;
- (5) eine (CASE-) **Plattform** (Prozesskoordination, Objektspeicher, Kommunikationsmechanismen) **für SEU**, die auch für andere verteilte Anwendungen genutzt werden kann;
- (6) eine **Meta-Umgebung** zum Bau von SEU (**Meta-CASE**).

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

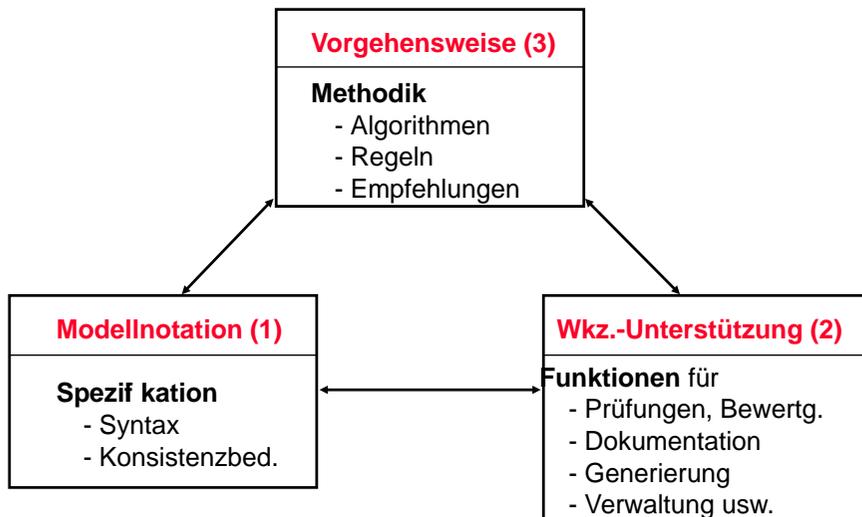
Prof. U. Aßmann, SEW 12

## 10.1.2 Aufbau und prinzipielle Funktion von Software-Entwicklungswerkzeugen

## Werkzeugfunktionen

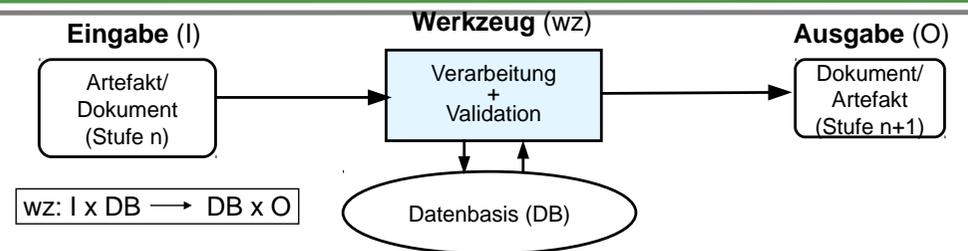
- ▶ Ursprünglich wurden nur einzelne grundlegende Komponenten der Software-Entwicklung wie Compiler, Editoren oder Testhilfen als Werkzeuge bezeichnet
- ▶ Im Laufe der Zeit kamen viele **spezialisierte** Entwicklungs- und Administrationswerkzeuge hinzu:
  - Herstellung und Verarbeitung von **Artefakten** oder **Dokumenten** (Prosa, Bilder, Diagramme, Modelle, Spezifikationen, formatierte Texte, Programme, Code)
    - --> **Modellnotation (1)**
  - Konsistenzprüfung auf Wohlgeformtheit von einzelnen Dokumenten und zusammengehörigen Dokumentenbeständen, Produktverwaltung während der Herstellung und Wartung
    - --> **Werkzeugprüfung/-Unterstützung (2)**
  - Unterstützung von Methoden und einzelner Entwicklungsschritte (Entwurf, Testen,...)
  - Unterstützung von Phasen- und Vorgehensmodellen
    - --> **Vorgehensweise/Methodik (3)**

## In Werkzeugen unterstützte Aspekte (Auch Modellaspekte der Basistechniken)

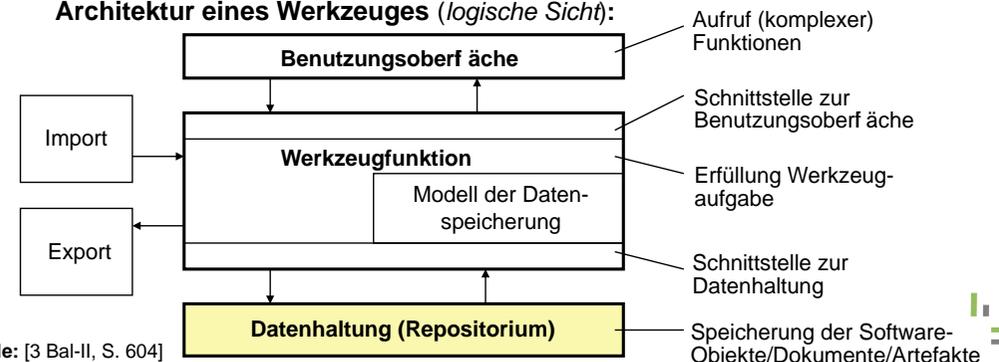


Quelle: nach Raasch, J.: Systementwicklung mit strukturierten Methoden; Hanser Verlag (2. Auflage) München 1992

## Werkzeug - Wirkungsschema

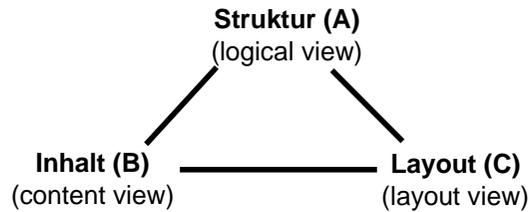


### Architektur eines Werkzeuges (logische Sicht):



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

## Aspekte von Dokumenten/Artefakten



**Struktur:** log. Einheiten, wie Gliederung, Überschriften, Fußnoten, Köpfe, Verweise  
kontextfreie Struktur

kontextsensitive Struktur (statische Semantik)

**Inhalt:** (Kapitel-)Text, Grafiken, Bilder, Bitmuster, elektron. Erscheinungsformen

**Layout:** Ausgabeanordnungen und -vorschriften für log. und inhaltliche Elemente

Standards, die Aspekte von Dokumenten trennen:

**SGML** = Standard Generalized Markup Language (Teilmenge ist HTML).

Legt in abstrakter Form die Struktur von Dokumenten fest.

**XML** = Extensible Markup Language - Strukturbeschr. mit DTD--> XML-Tags

Prof. U. Aßmann, SEW 17

## Dokumenttypen (Artefakte) der Softwareentwicklung

- ▶ Text
  - z.B. Anforderungsspezifikation, Entwurfsspezifikation, Programmbeschreibungen,...
- ▶ Grafik
  - z.B. Analyse- und Entwurfsspezifikation (UML-Diagramme), Programmstrukturen,...
  - komplexe visuelle Darstellungen in 2-D oder 3-D
- ▶ Code
  - z. B. Pseudocode, Codegerüste, Quellcode
- ▶ Tabellen
  - z.B. Relationen, Testfalltabellen

### Eigenschaften von Softwareentwicklungs-Dokumenten:

**Struktur** • **Struktur** meist vorgegeben (UML), Standardisierungsgrad wächst

• haben einen **Autor** (oder mehrere bei Gruppenarbeit),

• und verschiedene **Zielgruppen**,

**Inhalt** • sind **komplex** und **umfangreich**,

• müssen **präzise** sein (genaues Abbild des Originals, Formalisierung),

• durchlaufen einen **Entwicklungszyklus**,

**Layout** • sollen **lesefreundlich** und "**schön**" aussehen (Verständlichkeit),

• sind Gegenstand von **Reviews**,

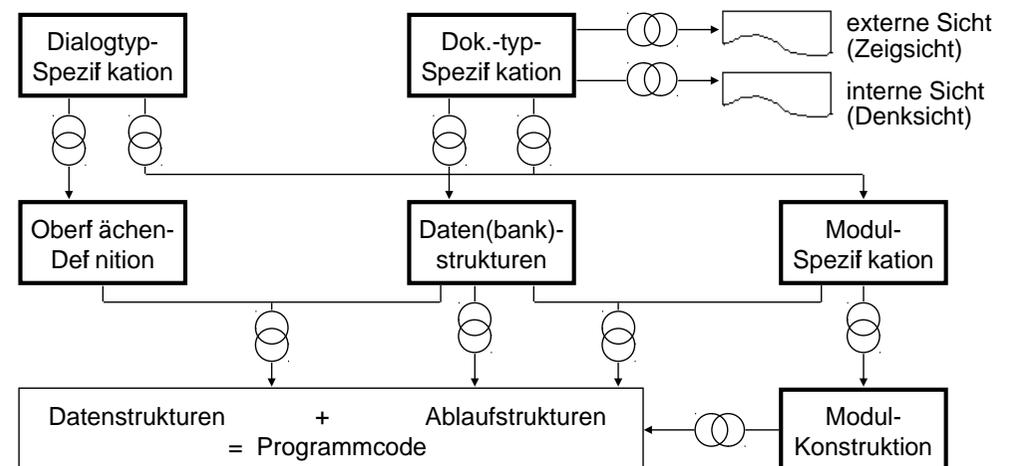
• werden **maschinell** erzeugt und geprüft (Validierung).

Quelle: Denert, E.: Dokumentenorientierte Software-Entwicklung; Informatik-Spektrum 16(1993) H. 3, S. 159 - 164 18

## Werkzeuggetriebene Softwareentwicklung

Dokumentenorientiert (artefaktor.)	transaktionsorientiert
Dokument als primäres Endprodukt steht ständig im Vordergrund	interaktive Entwicklung von Artefakten steht im Vordergrund
Entwickler schreibt Software	Entwickler zeichnet Software
typisch ist eindimensionaler Text, Grafiken werden eingeführt	typisch sind zweidimensionale Grafiken, Text eingefügt
Hauptwerkzeug ist universeller (Text-) Editor	mehrere methodenorientierte grafische Editoren
Methodik (Metamodell) flexibel	Methodik (Metamodell) durch CASE starr vorgegeben
ganzheitliche Denkweise ausgerichtet am Dokument	atomares Agieren der Elemente in der Datenbank
klare Abgrenzung der Verantwortlichkeit für Dokumente (evtl. Gruppenabstimmung)	Verantwortungsabgrenzung für grafische Elemente schwieriger

## Dokumententransformation durch Werkzeuge



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

## 10.2 Werkzeuggrundtypen - Klassen von CASE-Tools

SEW, © Prof. Uwe Aßmann

21

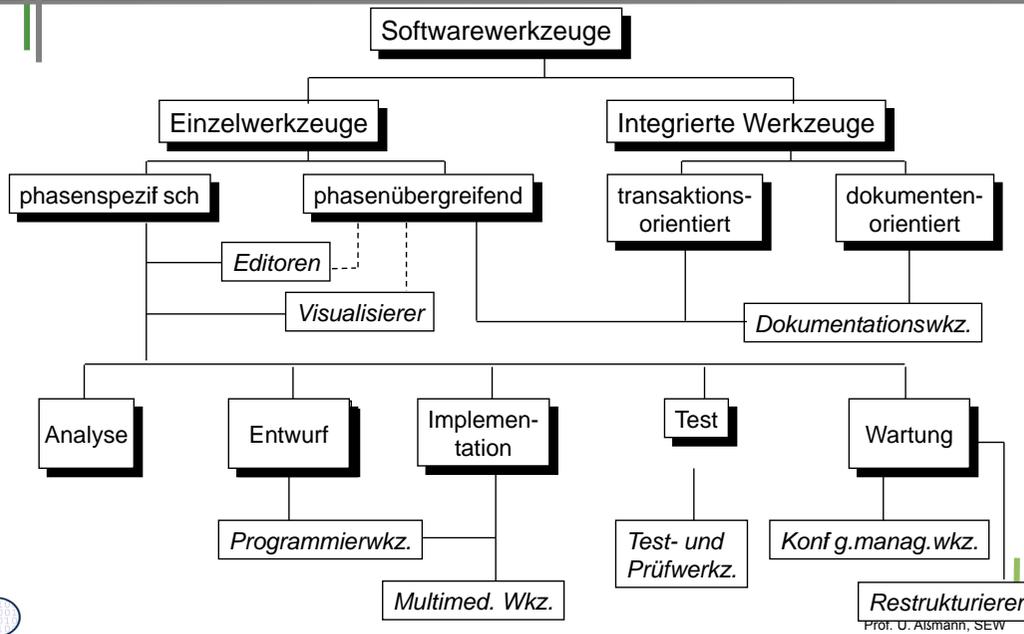
## Entwicklungsaufgaben und Werkzeuge

Vertikale, phasenspezifische Werkzeuge						Horizontale, phasenübergreifende Werkzeuge
Planung	Analyse	Entwurf	Konstr./Implemen.	Test	Wartung	
Dokumentation						
Zugriffssicherheit						
Produktverwaltung						
Konfigurationsmanagement						
Qualitätssicherung						
Projektmanagement						

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

22

## Eine Grobgliederung von Software-Entwicklungswerkzeugen



Prof. U. Aßmann, SEW

23

## Bespiele für phasenspezifische Wz: Editoren

<b>Aufgabe:</b>	Interaktive Bearbeitung (Erstellen, Modifizieren, Inspizieren) von Spezifikationen (Code, Text, Grafik, Bild usw.)
<b>Modell:</b>	die zu bearbeitenden Spezifikationen und Medienobjekte (Daten), die neutral (universelle Editoren) oder spezifisch (sprachorientiert, syntaxunterstützend) sein können.
<b>Sichten:</b>	Präsentation in der Regel statisch z. B. in Bildschirmfenstern, die alle Daten anzeigen
<b>Interaktion:</b>	Benutzereingaben (Tastendrucke, Mausclicks o.ä.) reagieren auf Eingabeteile (Texte, Programme, visuelle Darstellungen)
<b>Struktur des Modells:</b>	je nach Objektdefinition, Objektassoziationen, Methodenstruktur, Struktur des Vorgehens
<b>Zeitpunkt:</b>	in der Regel asynchron, synchron bei Teamarbeit
<b>Initiative:</b>	benutzerorientiert
<b>Arten:</b>	Texteditoren, Spracheditoren, Figureneditoren, Pixeleditoren

Prof. U. Aßmann, SEW

24

## Bespiele für phasenübergreifende Wz: Browser

<b>Aufgabe:</b>	Visualisierung externer und interner Strukturen zum selektiven Lesen großer Datenmengen, die in übersichtlicher und strukturierter Form aufbereitet werden (to browse = schmökern) auch mit Include-Funktionen
<b>Modell:</b>	der übersichtlich darzubietende Informationsraum z. B. in Form von Hierarchien, Objekten, Klassen, Methoden bzw. Dokumenten
<b>Sichten:</b>	Präsentation in der Regel statisch auch dynamisch in Bildschirmfenstern, die verständlichen Modellausschnitt anzeigen
<b>Interaktion:</b>	Navigieren und Explorieren in großen Strukturen auch unter Einschaltung von Auswahl- und Filterfunktionen
<b>Struktur des Modells:</b>	externe Objekt-/Methodenstruktur aber auch dynamische Methodenstrukturen
<b>Zeitpunkt:</b>	Visualisierung synchron während Programmlauf, seltener asynchron nach dem Programmlauf
<b>Arten:</b>	Datei-Browser, Symbol-Browser, Hierarchie-Browser, Netz-Browser, Schnittstellen-Browser

## Bespiele für phasenübergreifende Wz: Dokumentationswerkzeuge

<b>Aufgabe:</b>	Variable Aufbereitung von Auswertungsinformationen zur Erzeugung, Vervielfältigung, Ablage und Sammlung von Softwaredokumentationen (online-Handbüchern)
<b>Modell:</b>	Aufbau der Softwaredokumente nach Inhalt, Struktur, Layout, die variabel (Beschreibungssprachen) aufbereitet und ausgegeben werden müssen
<b>Sichten:</b>	Bereitstellung unterschiedlicher Sichten und Formate im Bildschirmfenster und WYSIWYG auf Drucker, Plotter
<b>Interaktion:</b>	Interaktive variable Aufbereitung im Dialog, Dokumentenproduktion im Batchbetrieb
<b>Struktur des Modells:</b>	je nach gewählten Firmenvorschriften, Standards, z. B. DOD-STD-2167A (Defense System Software Development)
<b>Zeitpunkt:</b>	in der Regel asynchron zeitversetzt
<b>Initiative:</b>	benutzerorientiert in der Aufbereitung und bei Abarbeitung systemgesteuert
<b>Arten:</b>	Textverarbeitung (Word), Desktop Publishing (FrameMaker), Dokumentengeneratoren

## Bespiele für phasenspezifische Wz: Generatoren

<b>Aufgabe:</b>	Transformation von grafischen und/oder textuellen (Entwurfs-)Spezifikationen in eine höhere Formalisierung wie Code, Codegrüste, Programmteile oder ganze Programme
<b>Modell:</b>	Syntax und Semantik der Eingabesprache, aus der über feste Abbildungsregeln in die gewünschten Ausgabespezifikationen zu erzeugen sind
<b>Sichten:</b>	statische Eingabeströme, die über Steuerparameter angepasst werden können
<b>Interaktion:</b>	Visualisierung von Fehlersituationen mit der Möglichkeit des direkten Eingreifens im Dialogbetrieb
<b>Struktur des Modells:</b>	je nach Definition der Eingabesprache und der daraus zu generierenden Ausgabe-Objekte
<b>Zeitpunkt:</b>	in der Regel asynchron, Ausgabe versetzt nach Korrektur
<b>Initiative:</b>	Systeminitiative, nur benutzerorientierte Eingabeaufbereitung.
<b>Arten:</b>	Programmgeneratoren, Masken-Generatoren, Test(fall-)Generatoren, Report-Generatoren

## Bespiele für phasenspezifische Wz: Debugger

<b>Aufgabe:</b>	Steuerung, Anzeige und Protokollierung der Zustände zu testender Programme in lesbarer Form
<b>Modell:</b>	die Struktur des Testprogramms mit definierten Mengen fester sowie variabler Haltepunkte
<b>Sichten:</b>	Anzeige der momentanen Stelle im Programm, des Inhalts lokaler und globaler Variabler und in Ausführung befindlicher Prozeduren durch Transformation des Maschinen-zustandes in den zugeh. Zustand des Quellprogramms
<b>Interaktion:</b>	Benutzereingaben zur Reaktion auf online Debugger-Ausgaben, Manipulation in Programm-(Objekt-)Strukturen
<b>Struktur des Modells:</b>	gebildet durch Referenzinformationen (Namen, Typen, Adressen von Variablen und Prozeduren)
<b>Zeitpunkt:</b>	synchron bei interaktiver Abarbeitung ( <i>Tracing Debugger</i> ) und asynchron bei <i>Post-Mortem</i> Aufruf
<b>Initiative:</b>	benutzerinitiiert bei <i>Tracing</i> und systeminitiiert bei <i>Post-M.</i>
<b>Arten:</b>	absolute Debugger (Masch.-Niveau), symbolische Debugger (Source-Code), grafische Debugger

## Klassen von CASE-Tools (1)

Klasse	Subklassen	Grundfunktion
<b>Editor</b>	Text-Editor Graphischer Editor	Erstellen und Modifizieren aller Arten von Spezifikationen und Medienobjekten
<b>Visualisierer</b>	Browser Viewer Inspektor	Visualisierung externer Strukturen und multimedialer Objekte (Hierarchien, Aufrufe bzw. Dokumente). Interaktion mit anderen Werkzeugen.
<b>Programmierwerkzeuge</b>	(Cross-)Assembler (Cross-)Compiler Interpreter Linker Generator Restrukturier	Transformation textueller (und visueller) Programmspezifikationen in einen Zwischencode oder Maschinencode, der unmittelbar interpretiert oder aus dem Objektcode ausgeführt wird. Analyse der Syntax (und der Semantik). Erzeugung v. Codegerüsten aus sem. Spezifikation u. umgekehrt.
<b>Test- und Prüfwerkzeuge</b>	Manipulatoren Debugger Tracer Stat. Analyzers Dyn. Analyzers Comperator	Syntaktische Überprüfung von Spezifikationen u. Programmen. Manipulation der Eingangsbedingungen und Festlegung des Testablaufes. Vergleich mit Sollbedingungen. Auswertung und Visualisierung der Ergebnisse nach unterschiedlichen Kriterien, Metriken

Quelle: Fugetta, A.: A Classification of CASE Technology; Computer 26(1993) H. 12, S. 25-38

Prof. U. Aßmann, SEW

29

## Klassen von CASE-Tools (2)

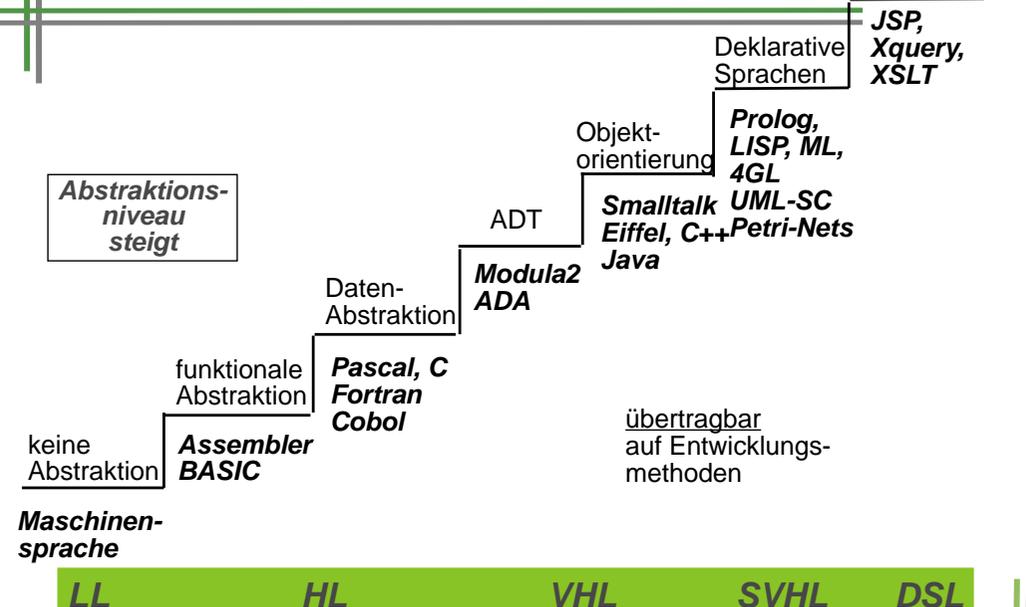
Klasse	Subklassen	Grundfunktion
<b>Test- und Prüfwerkzeuge</b>	Simulator Testfall-Generator Testverwaltungs.werkz.	Interpretation der Ausführung von Spezifikation. Erzeugen von Testdaten aus der Progr.spezif. k. Verwaltung aller Testresultate, Testreports
<b>Konfigurationsmanag.-Werkzeuge</b>	Repositories Versionsverwalter Konfigurationsbuilder Produktverwalter	Ablage und Verwaltung sowohl anwendungsbezogener als auch multimedialer Objekte. Überwachung der Versions- und Konfigurationsbildung. Navigations- und Wiederaufnahmefunktion.
<b>Dokumentationswerkzeuge</b>	Textverarbeitung Transformatoren Desktop Publishing Hypertextsysteme Spreadsheets	Variable Aufbereitung von Auswertungsinformationen nach Struktur, Inhalt und Layout. Erzeugung von (online-)Handbüchern, Projektdokumentationen und Tutorials in unterschiedlichen Formaten.
<b>Multimediale Werkzeuge</b>	User Interface Builder Autorensysteme Synchronisatoren	Bildung und Integration dynamischer und multimedialer Objekte in einer Applikation bestehend aus Präsentation und Interaktionssteuerung.
<b>Wartungswkz.</b>	Änderungskontrolle Fehlerbehebungshilf.	Überwachung aller Programmänderungen, Koordinierung der Zugriffsberechtigungen

Prof. U. Aßmann, SEW

30

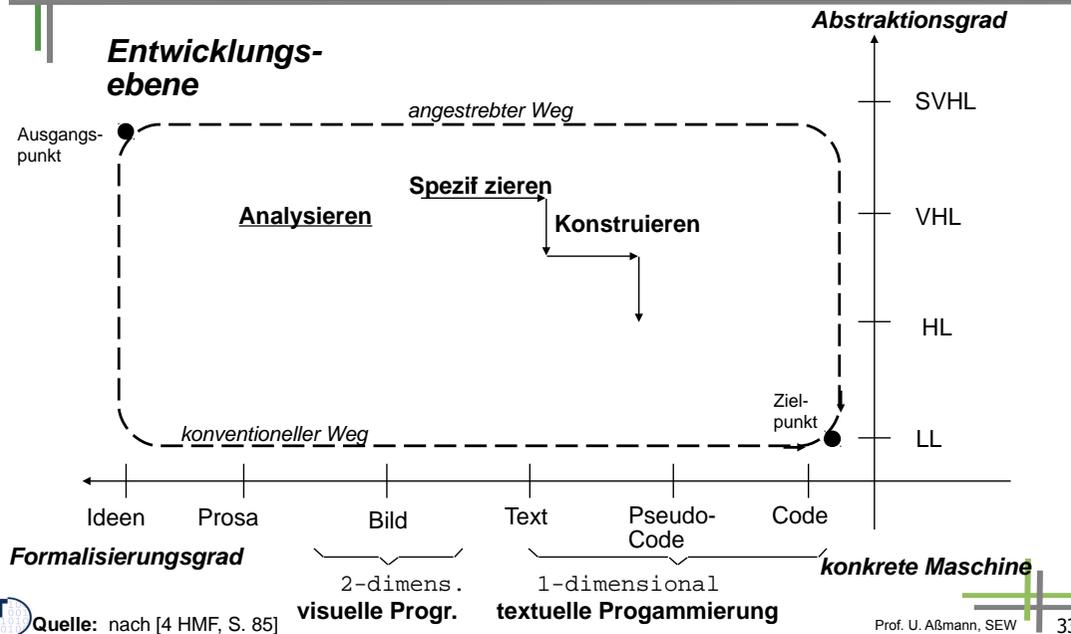
## 10.3 Werkzeug-Landschaft nach Hesse

## Evolution der Programmiertechniken

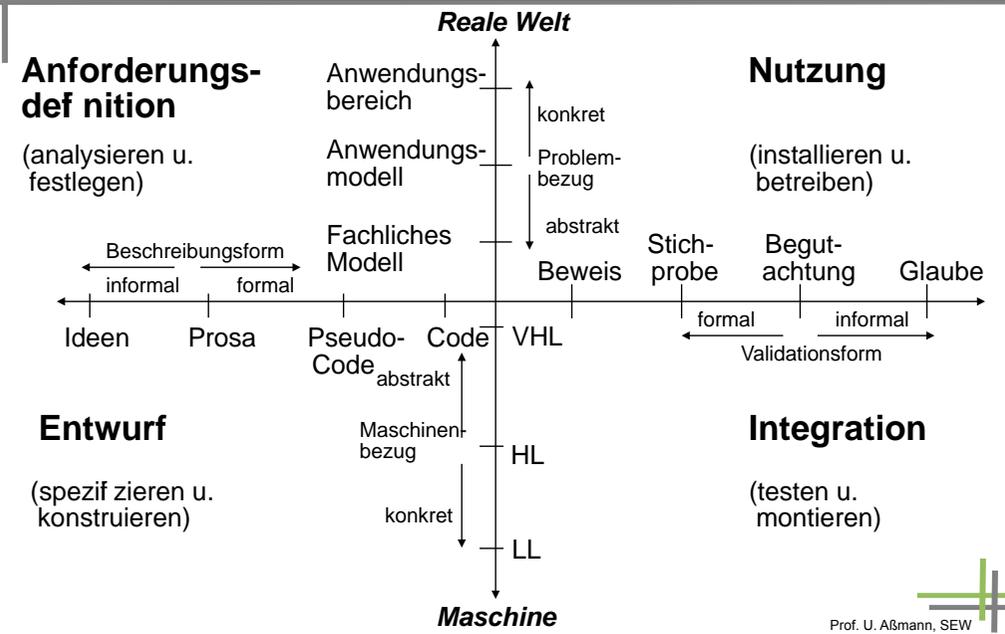


Domänen  
spezifische  
Sprachen

# Abstraktion der Softwareentwicklung



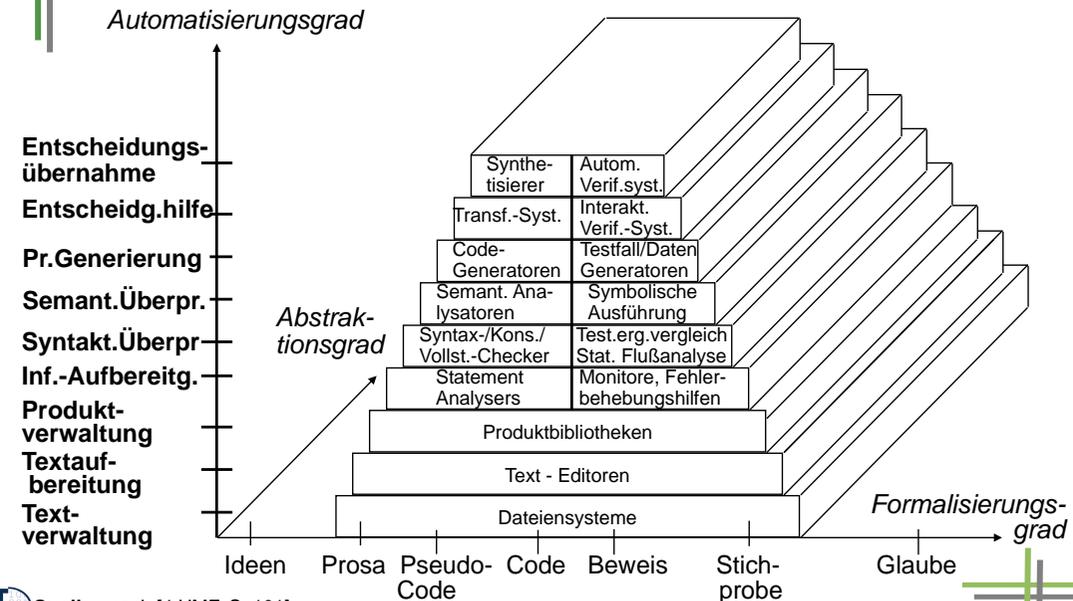
# Software-Entwicklungsquadranten



# Automatisierungsgrad von Werkzeugen

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 Verifikation
7	(Produkt-)Generierung	Automatische Erzeugung von Codegerüsten (Programmen) aus Entwürfen und Testfällen/Testdaten aus der Anforderungsspezifikation
6	Semantische Überprüfung	Analyse z.B. des kontext-sensitiven Teils formaler Spezifikationen und andere die Programmausführung betreffende Inhalte
5	Syntaktische Überprüfung	Vollständige synt. Überprüfung formaler Spezifikationen durch „Syntax-Checker“, Parser, Flussanalysen usw.
4	Informations-Aufbereitung	Syntaktische Analyse von bestimmten formal-sprachlichen Informationen, Ausgabe von Inkonsistenzen, Fehlern, Querbezügen
3	Produktverwaltung	Manipulieren und Verwalten von wohldefinierten „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



## Verwendung typischer CASE-Tools

Entwicklungsphase	Spezialisierte Werkzeuge
Anforderungs-Analyse	User Interface Prototyping Tools OOA-Tools (Use Case, Object, Class) Information Modeling Tools Structured Analysis Tools Real Time Modeling Tools
Entwurf	OOD-Tools (Class Modeling) Daten-Modellierungswerkzeuge Modul/Package Specification Tools
Implementation Wartung	Codegeneratoren Compiler/Interpreter Symbolic Debugger Smart Text Editor Execution Profilers Konfigurationsmanagementsysteme

Quelle: nach [8, S.22]

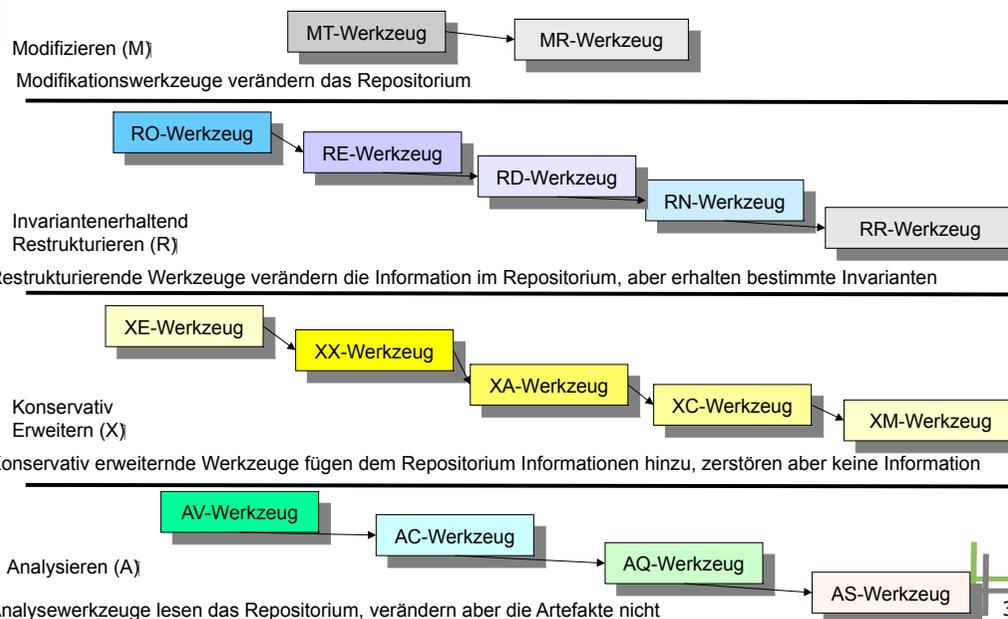
Prof. U. Aßmann, SEW 37

## 10.4 Einführung in die Effektkategorien für Werkzeuge

SEW, © Prof. Uwe Aßmann

38

## Effektkategorien ("Blutgruppen") für Werkzeuge



Analysewerkzeuge lesen das Repository, verändern aber die Artefakte nicht

39

## 10.5 Der Graph-Logik-Isomorphismus

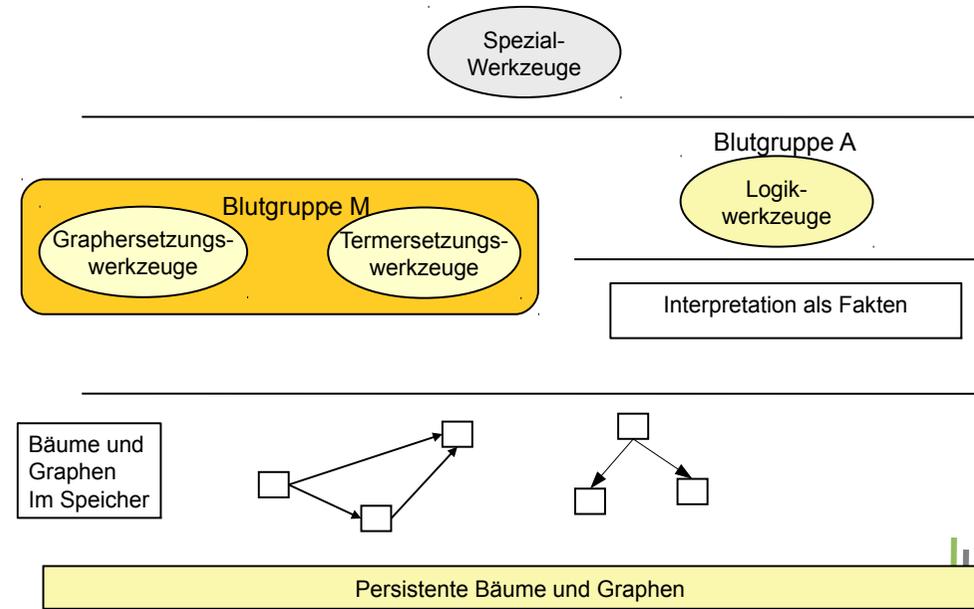
SEW, © Prof. Uwe Aßmann

40

## Der Graph-Logik-Isomorphismus

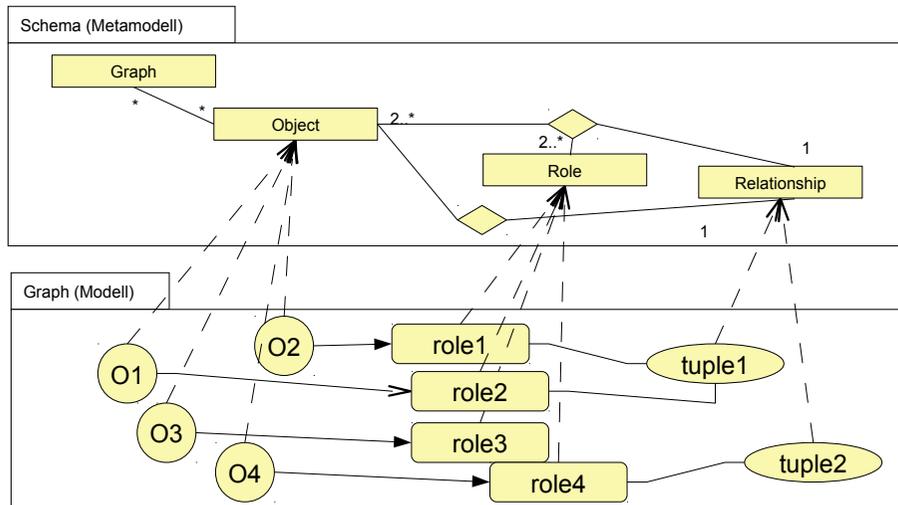
- ▶ Jeder Graph kann als Faktenbasis einer Logikmaschine abgelegt werden.
- ▶ Jede Faktenbasis kann als Graph interpretiert werden
  - binär: Graph
  - n-är: Hypergraph
- ▶ Logikmaschinen und Graphtransformatiions-Werkzeuge können zu guten Teilen ausgetauscht werden

## SEU mit Ersetzungs- und Logik-Werkzeugen



## Typisierte Graphen (Modelle und Metamodelle)

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



End