43 Verfeinerung von Lebenszyklen - Geschichtete Interpretierer (Automaten)

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

Version 13-1.0, 06.07.13

1) Geschichtete lebenszyklen und Interpretierer
2) Anwendungen

Literatur

- Walter F. Tichy. 1992. Programming-in-the-large: past, present, and future. In Proceedings of the 14th international conference on Software engineering (ICSE '92). ACM, New York, NY, USA, 362-367. DOI=10.1145/143062.143153 http://doi.acm.org/10.1145/143062.143153

Wdh.: Punktweise vs querschneidende Verfeinerung

- Punktweise Verfeinerung:
  - Verfeinerung von Lebenszyklen, d.h. Objekten, Operationen und Attributen
  - Arbeit jeweils an *einem* Punkt der Spezifikation
- Querschneidende Verfeinerung:
  - Arbeit an *mehreren* Punkten der Spezifikation
  - Kapselung des Querschnittsverhaltens in einer Kollaboration (Konnektor)

43.1 Geschichtete Lebenszyklen

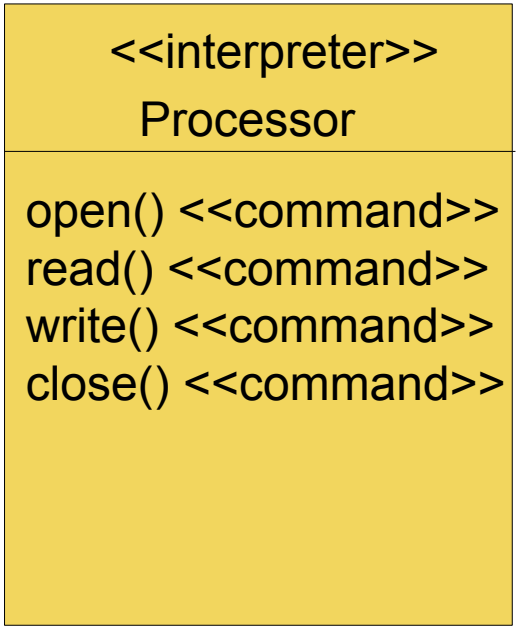
Geschichtete Steuerungsmaschinen (Abstrakte Maschinen, Layered Abstract Machines, Layered Interpreters)

Architekturstil “Geschichtete Lebenszyklen”

- Der Architekturstil *Geschichtete Lebenszyklen* (geschichtete Steuerungsmaschinen, abstrakte Maschinen, Layered Abstract Machines)
  - benutzt punktweise Verfeinerung, um höher liegende abstrakte Maschinen (Tools, Interpretierer) mit ausdrucksstarken Kommandosprachen in niederer liegende abstrakte Maschinen abzubilden
  - gliedert die Anwendungslogik-Schicht in der 3-Schichten-Architektur in Schichten
- Dominant bei interaktiven Anwendungen
  - Büroautomation (office systems)
  - Editoren
  - Formular-basierte Anwendungen, auch Web
- Auch für batch-Systeme (ohne Interaktion)
  - Auftragsbearbeitung (Order processing)
  - Transaktionsverarbeitung (OLTP, online transaction processing)

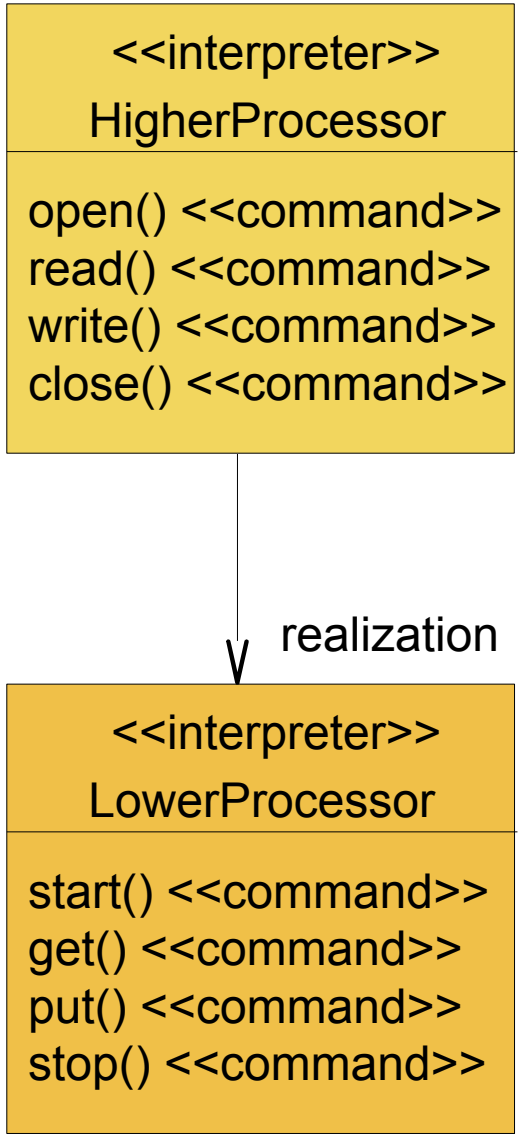
Layered Abstract Machines (Layered Interpreters)

- Eine *abstrakte Maschine (Interpreter, abstract machine, interpreter, Tool)* besteht aus
  - Einer Menge von Operationen und gekapselten Daten
  - Wird realisiert auf einer niederer liegenden abstrakten Maschine (verborgen)
- Wenn die abstrakte Maschine mit einem endlichen Automat (statechart) als Lebenszyklus versehen wird, sprechen wir von einer *Steuerungsmaschine* (siehe zuvor)
  - Siehe Entwurfsmuster Command and Interpreter (Gamma-Buch”)



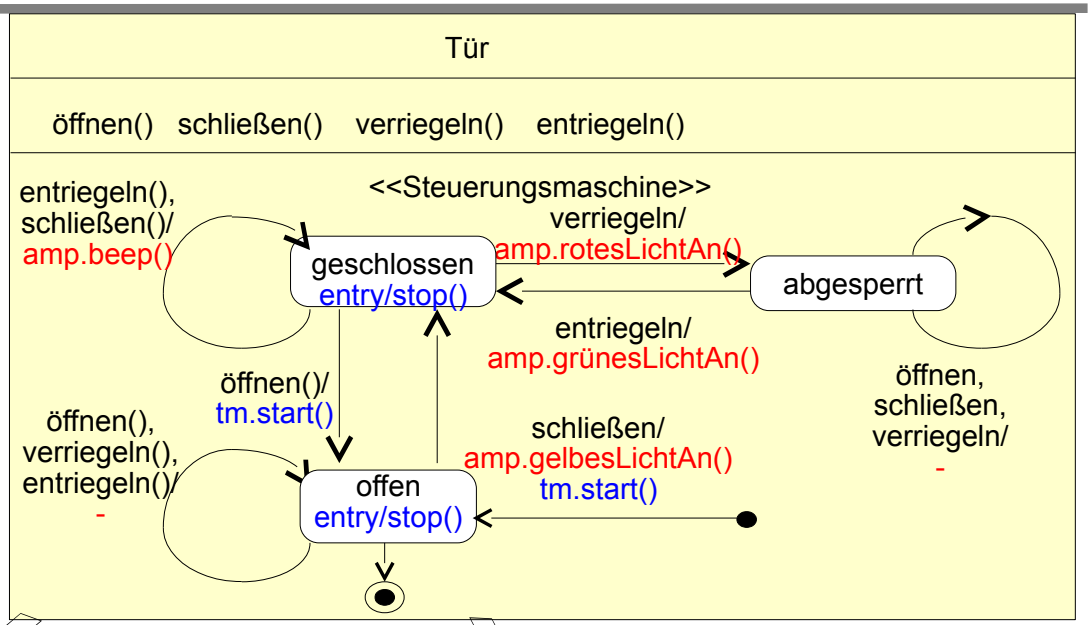
Layered Abstract Machines (Layered Interpreters, Layered Automata)

- Die Relies-On-Relation zwischen abstrakten Maschinen muss zyklenfrei sein

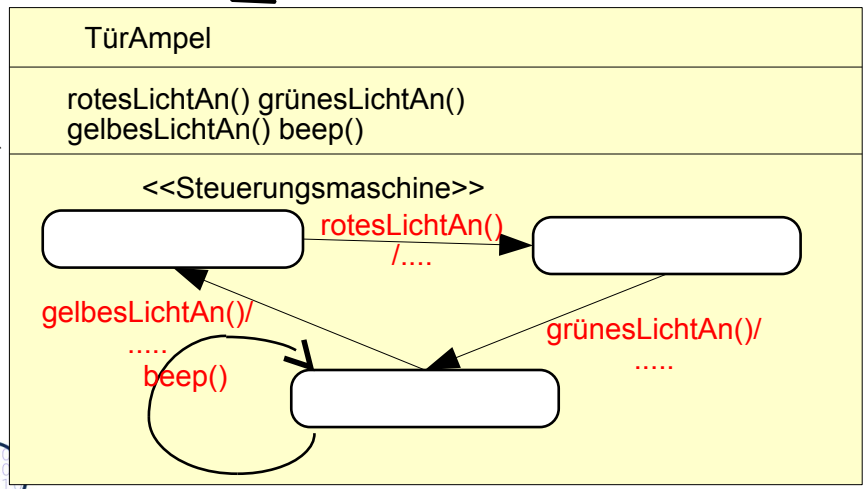


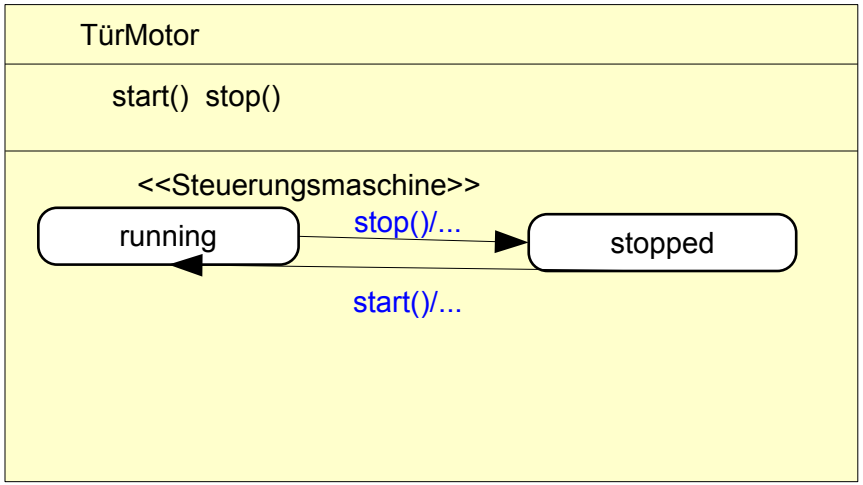
Realisierung von Interpretern mit Steuerungsmaschinen

- Verhalten von Interpretern kann durch Steuerungsmaschinen beschrieben werden können
- Interpreter (Steuerungs-maschinen) auf oberen Schichten können Interpreter auf unteren Schichten steuern



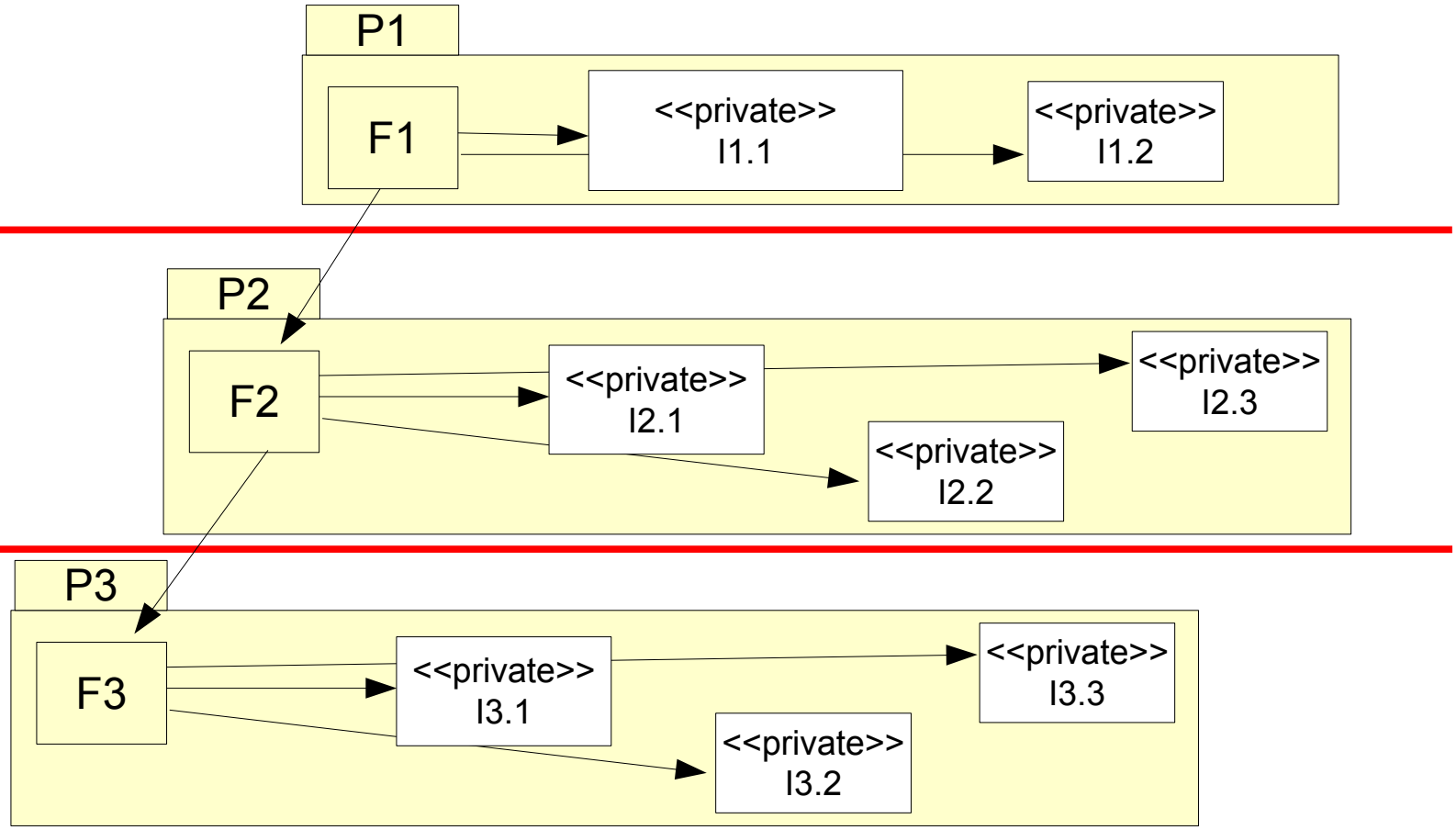
Schichtgrenze



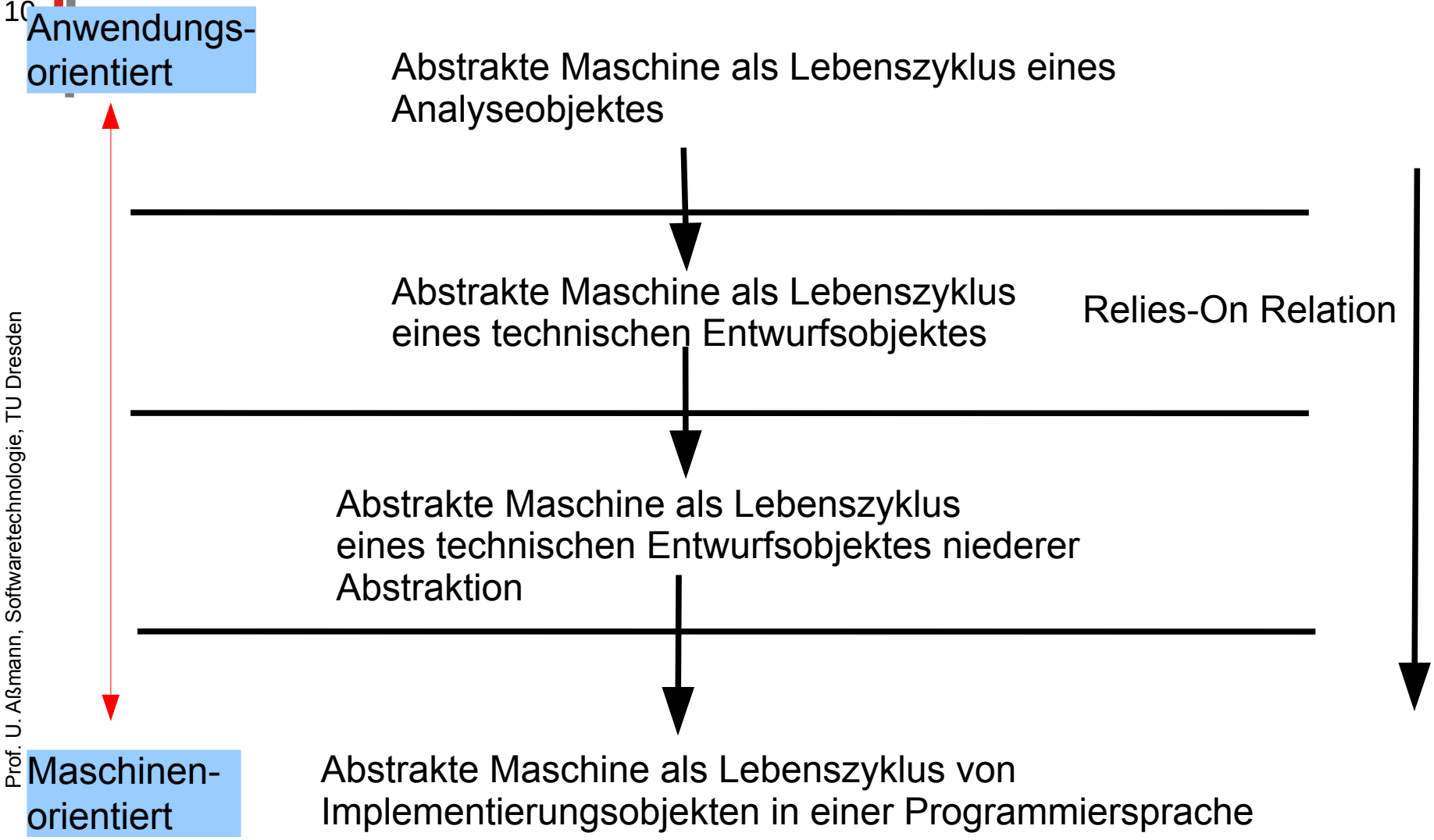


Realisierung von Interpretern durch Steuerungsmaschinen

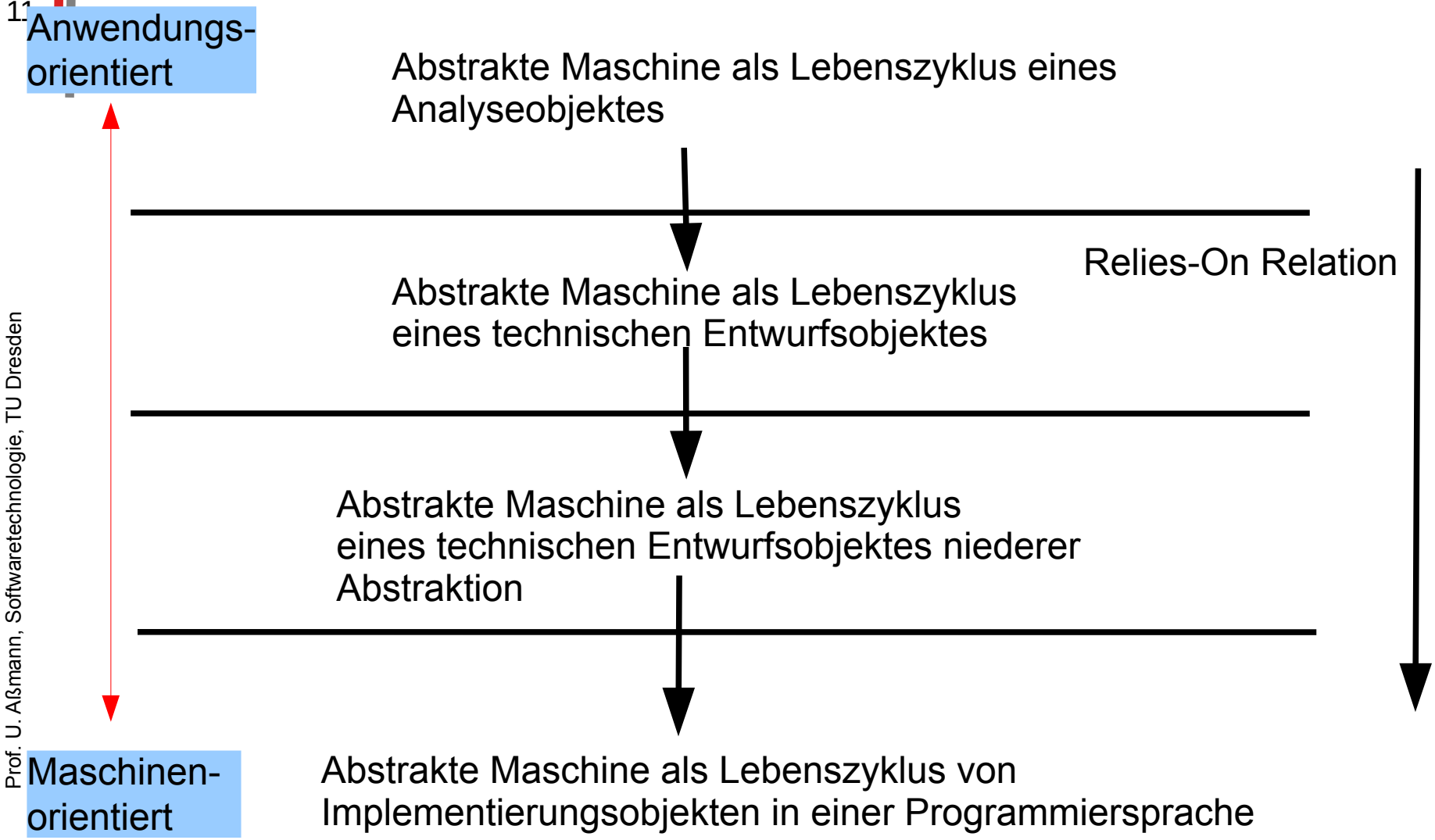
- Gelingt es, die Anwendung durch Schichten von Steuerungsmaschinen zu beschreiben, liegt ein sehr stark strukturierte Variante von Geschichteten Abstrakten Maschinen vor: *Geschichtete Steuerungsmaschinen (layered behavioral machines)*
- Diese können durchaus als Fassadenklassen von Paketen dienen, die das ganze Paket steuern



Geschichtete Steuerungsmaschinen (Layered Abstract Machines)

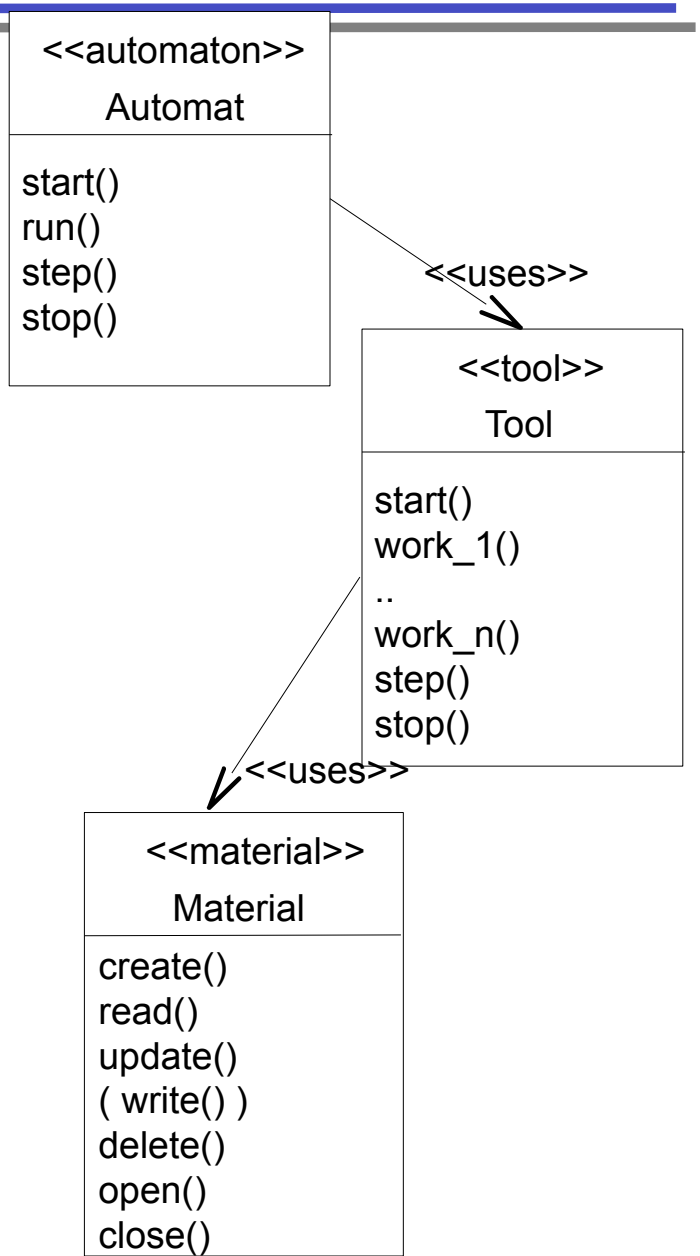


Verfeinerung mit geschichteten Steuerungsmaschinen (top-down)



3 Arten von Lebenszyklen: Tools, Materialien, Automaten

- **Werkzeuge (tools)**
  - Aktiv
  - Vom Benutzer oder von Automaten aus angesteuerbar
- **Materialien (materials)**
  - Passiv; nur über ein Tool benutzbar
  - In die Datenbank (E- und D- Schicht)
  - Gehorchen der CRUD-Schnittstelle (create, read, update, delete)
- **Automat (Interpretierer, automaton, workflow, interpreter)**
  - Arbeitsfluss; Programmsteuerung
  - Steuert Werkzeuge an, die auf Materialien arbeiten
  - Gleiche Schnittstelle wie tool



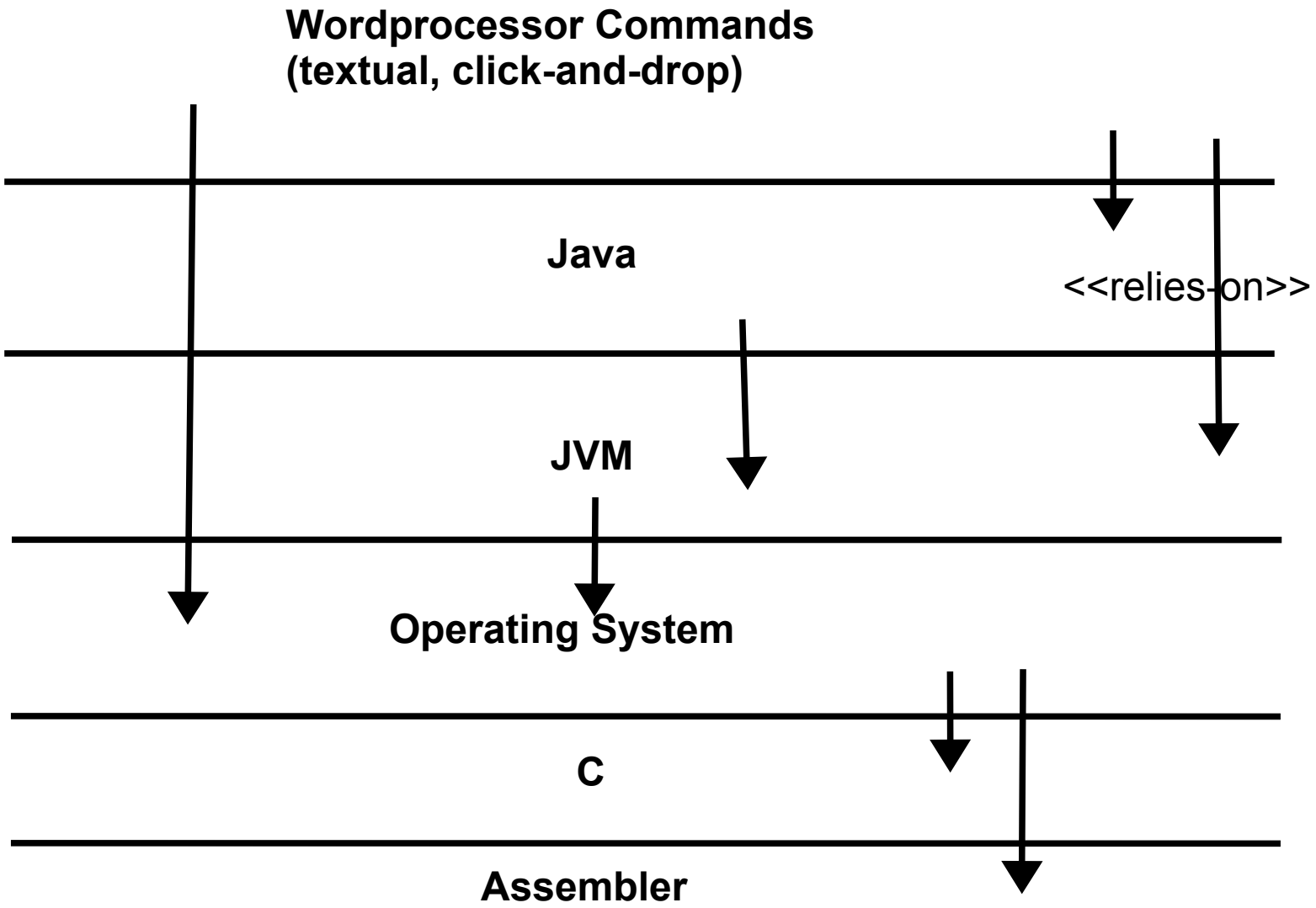
43.2 Anwendung von geschichten abstrakten Maschinen

Der ganze Computer ist eine einzige geschichtete abstrakte Maschine

- Befehle höherer Ebenen werden auf Befehle niederer Ebenen abgebildet

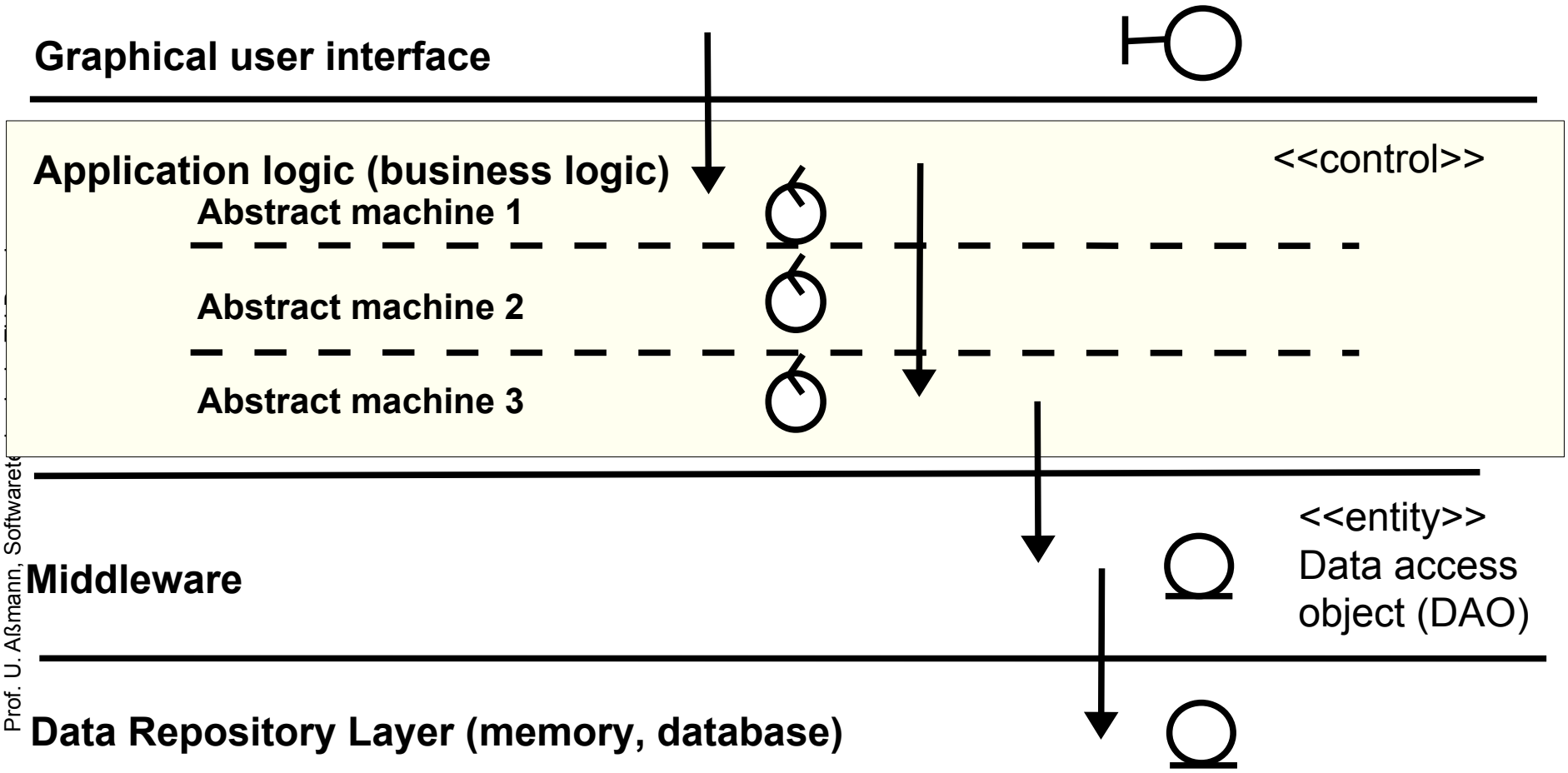
Kommandosprache in der GUI	**Apple Automator**
Domänenspezifische Sprache	**Mathlab, Simulink**
Specification language	**Prolog-Interpreter**
High level programming language	**shell-Interpreter, VDM**
Intermediate language	**JVM-Interpreter, emacs Lisp Code, .NET-VM**
Assembler	
Machine code **Kernel interface**	**Chip simulator** **OS**
Microcode	**Microcode interpreter**
Gates	
Physics	
?	

Beispiel: Text-Programm



Strukturierung der Anwendungslogik in Schichten

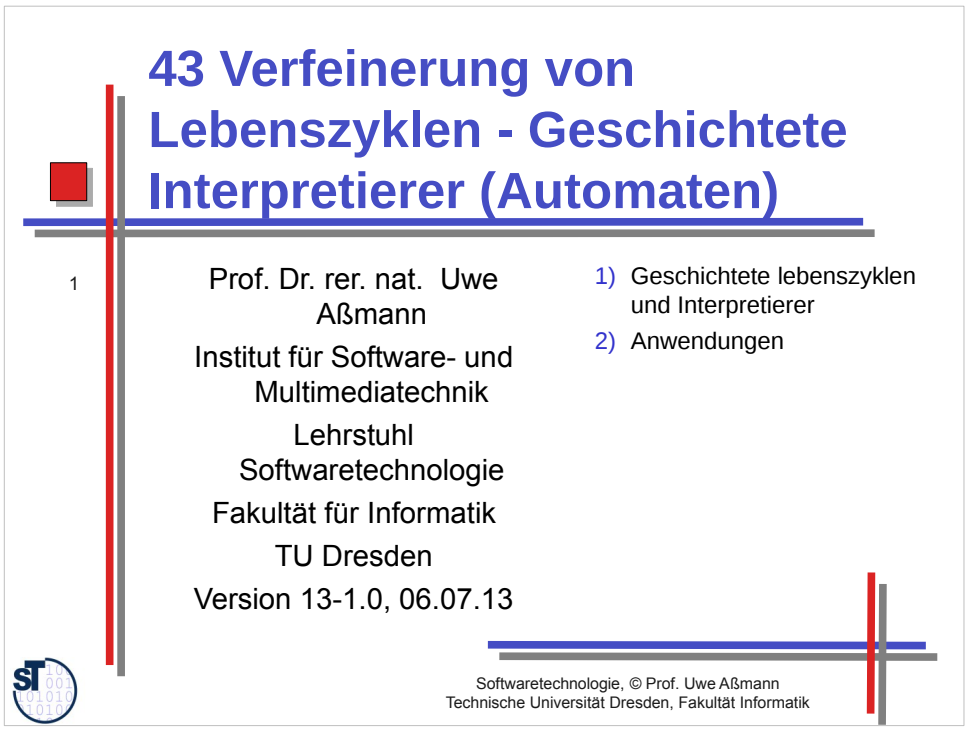
- Punktweise Verfeinerung der Anwendungslogik in Schichten



Entwurf mit geschichteten Abstrakten Maschinen

- Nutze BCED-Architekturstil
- Identifiziere abstrakte Maschinen in der Control-Schicht
  - Ordne sie in Schichten an
  - Denke über ihr “Schichtengeheimnis” nach
  - Kapsele eine Schicht hinter eine Fassade (und in ein Paket)
  - Statte die Fassade mit einer Steuerungsmaschine aus
- Vorteile:
  - Einfachheit
  - Hohe Kohäsion, niedrige Kopplung
  - Gute Austauschbarkeit
  - Gute Variierbarkeit
- Gehe für die anderen Schichten ähnlich vor

The End



- hier mussen Workflow-Architekturen hinein, Business Process Modeling

Literatur

- Walter F. Tichy. 1992. Programming-in-the-large: past, present, and future. In Proceedings of the 14th international conference on Software engineering (ICSE '92). ACM, New York, NY, USA, 362-367. DOI=10.1145/143062.143153 http://doi.acm.org/10.1145/143062.143153

Wdh.: Punktweise vs querschneidende Verfeinerung

- Punktweise Verfeinerung:
  - Verfeinerung von Lebenszyklen, d.h. Objekten, Operationen und Attributen
  - Arbeit jeweils an *einem* Punkt der Spezifikation
- Querschneidende Verfeinerung:
  - Arbeit an *mehreren* Punkten der Spezifikation
  - Kapselung des Querschnittsverhaltens in einer Kollaboration (Konnektor)

43.1 Geschichtete Lebenszyklen

Geschichtete Steuerungsmaschinen (Abstrakte Maschinen, Layered Abstract Machines, Layered Interpreters)

hichtete

ebenszyklen (geschichtete
Maschinen, Layered Abstract Machines)

um höher liegende abstrakte Maschinen
ksstarken Kommandosprachen in niederer
zubilden

nicht in der 3-Schichten-Architektur in Schich

lungen

auch Web

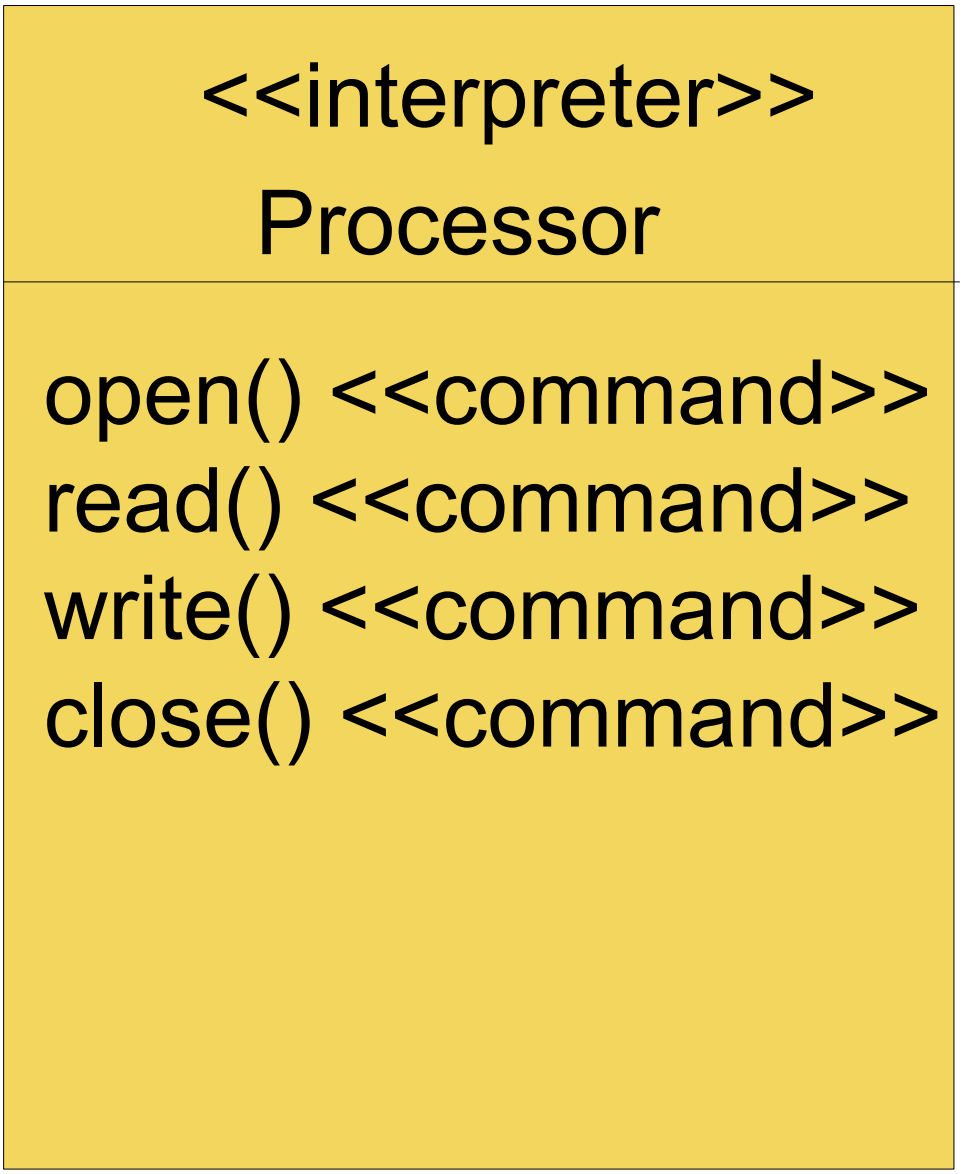
eraktion)

essing)

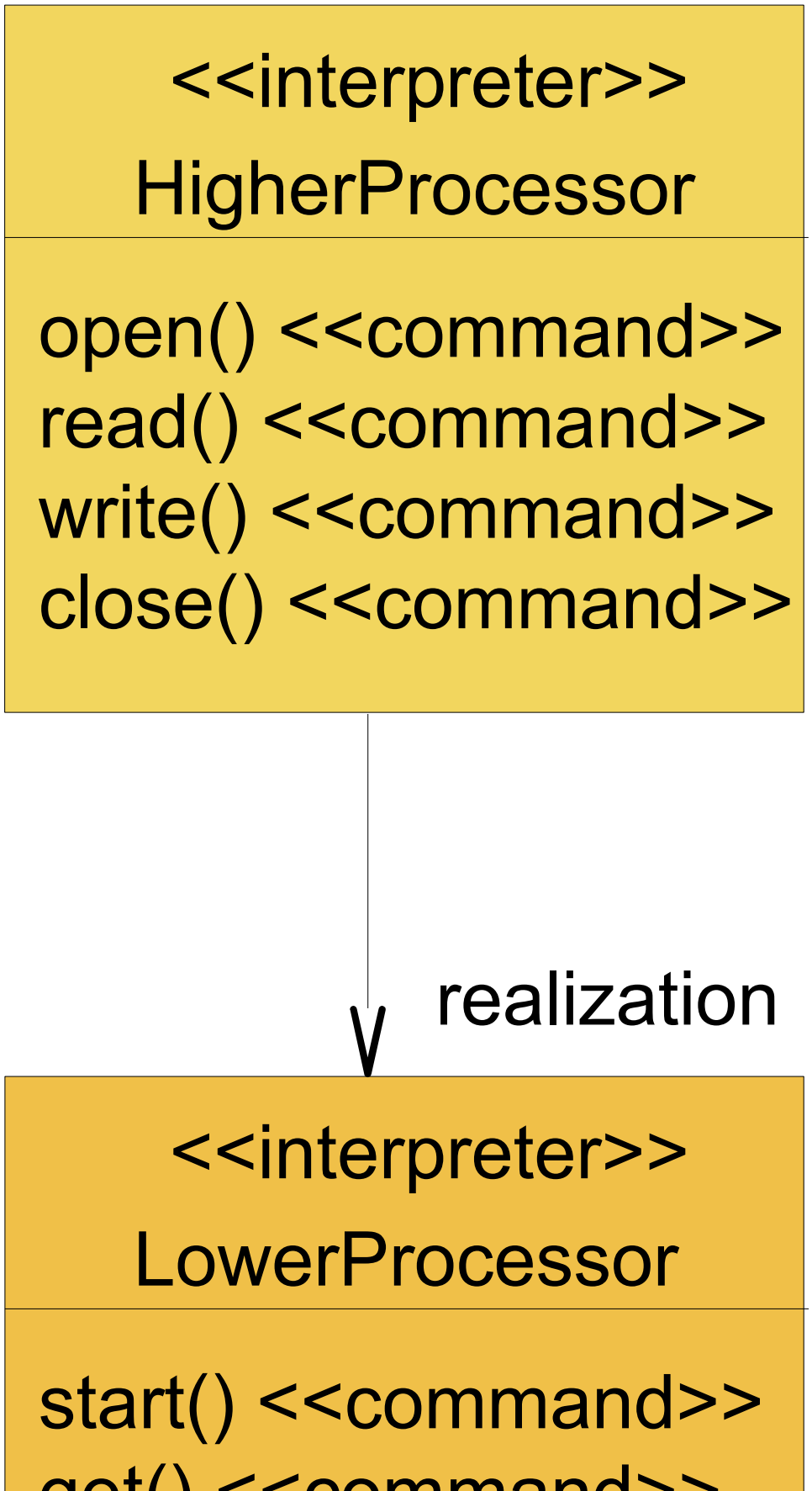
online transaction processing)

chines (Layered

d

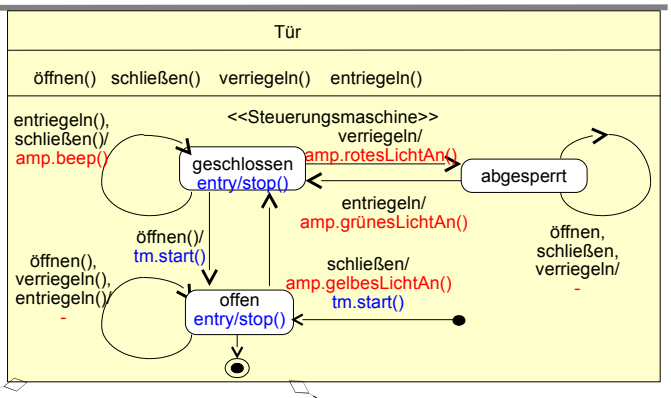


chines (Layered
l Automata)

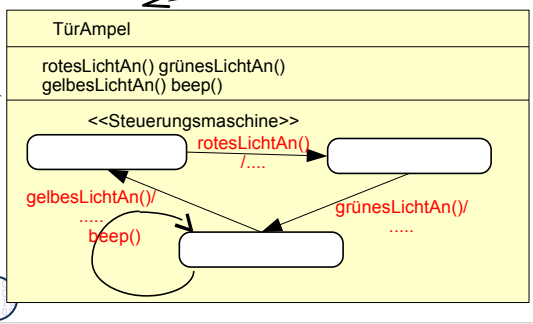


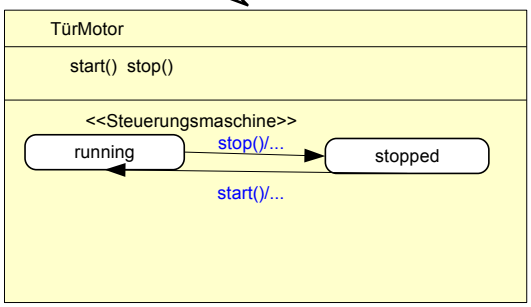
Realisierung von Interpretern mit Steuerungsmaschinen

- Verhalten von Interpretern kann durch Steuerungsmaschinen beschrieben werden können
- Interpreter (Steuerungs-maschinen) auf oberen Schichten können Interpreter auf unteren Schichten steuern



Schichtgrenze

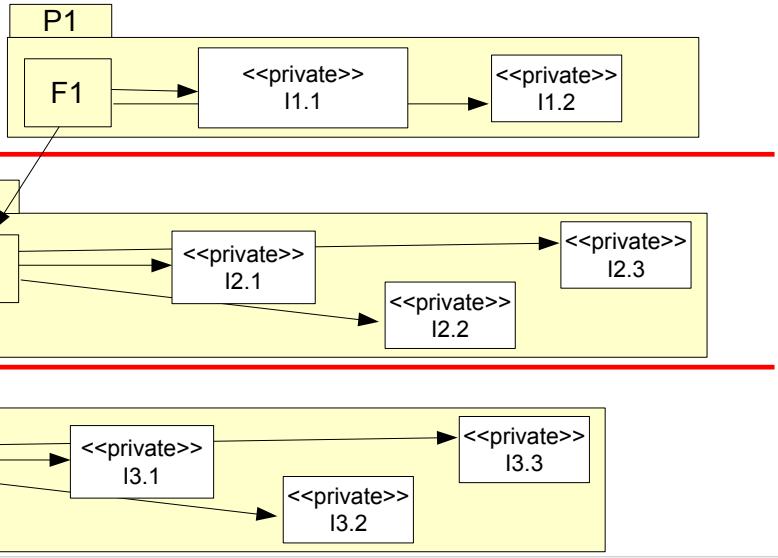




Prof. U. Aßmann, Softwaretechnologie, TU Dresden

Realisierung von Interpretern durch Steuerungsmaschinen

- Gelingt es, die Anwendung durch Schichten von Steuerungsmaschinen zu beschreiben, liegt ein sehr stark strukturierte Variante von Geschichteten Abstrakten Maschinen vor: *Geschichtete Steuerungsmaschinen (layered behavioral machines)*
- Diese können durchaus als Fassadenklassen von Paketen dienen, die das ganze Paket steuern



ungsmaschinen
achines)

als Lebenszyklus eines

als Lebenszyklus
ntwurfsobjektes

Relies-On Relation

Lebenszyklus
urfsobjektes niederer

chichteten
en (top-down)

als Lebenszyklus eines

Relies-On Relation

als Lebenszyklus
ntwurfsobjektes

Lebenszyklus
urfsobjektes niederer

- **Werkzeuge (tools)**
  - Aktiv
  - Vom Benutzer oder von Automate angesteuerbar
- **Materialien (materials)**
  - Passiv; nur über ein Tool benutzb
  - In die Datenbank (E- und D- Schi
  - Gehorchen der CRUD-Schnittstel (create, read, update, delete)
- **Automat (Interpretierer, automat workflow, interpreter)**
  - Arbeitsfluss; Programmsteuerung
  - Steuert Werkzeuge an, die auf Ma arbeiten
  - Gleiche Schnittstelle wie tool

43.2 Anwendung von geschichten abstrakten Maschinen

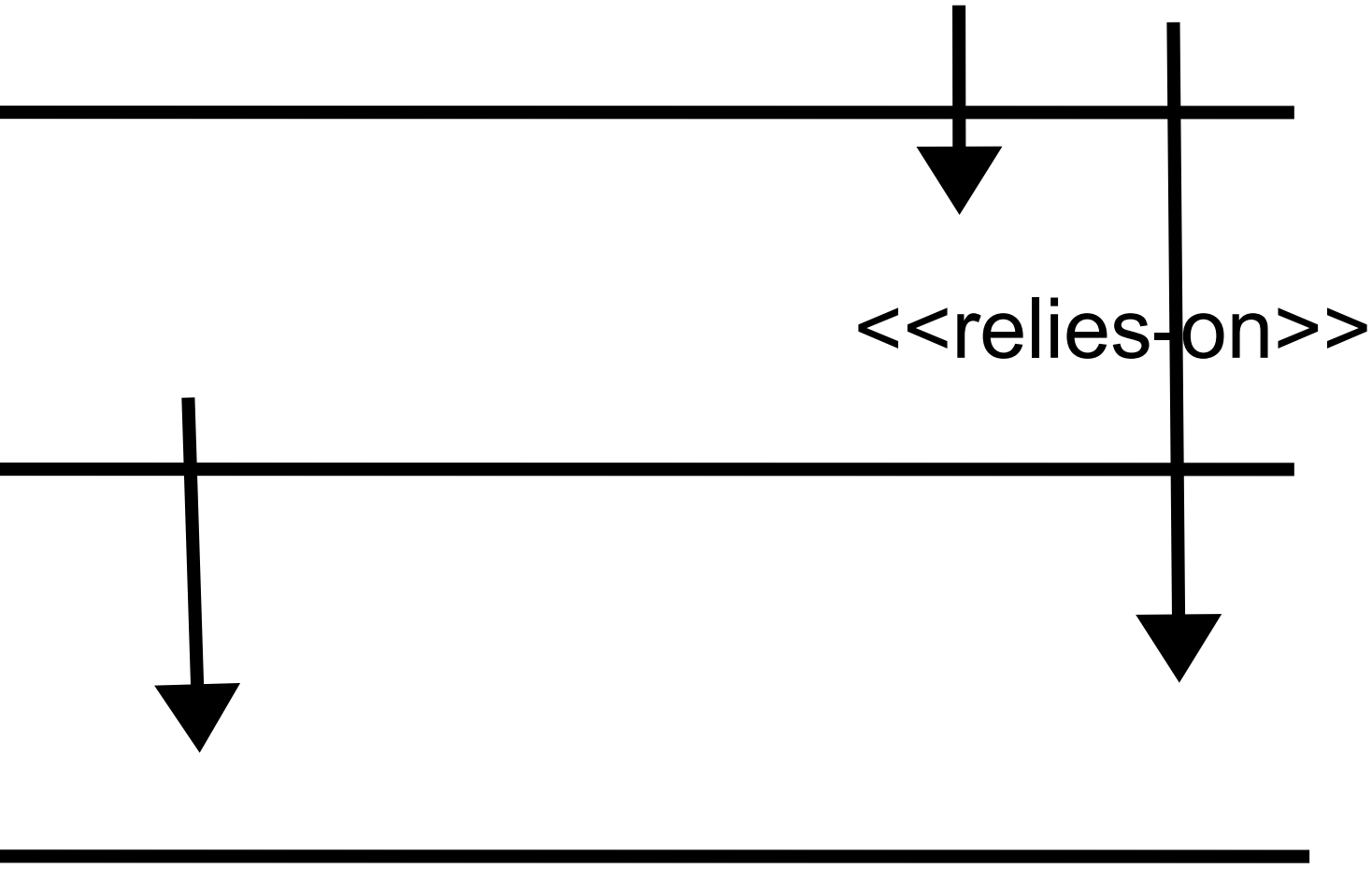
Der ganze Computer ist eine einzige geschichtete abstrakte Maschine

- Befehle höherer Ebenen werden auf Befehle niederer Ebenen abgebildet

Kommandosprache in der GUI	Apple Automator
Domänenspezifische Sprache	Mathlab, Simulink
Specification language	Prolog-Interpreter
High level programming language	shell-Interpreter, VDM
Intermediate language	JVM-Interpreter, emacs Lisp Code, .NET-VM
Assembler	
Machine code Kernel interface	Chip simulator OS
Microcode	Microcode interpreter
Gates	
Physics	
?	

amm

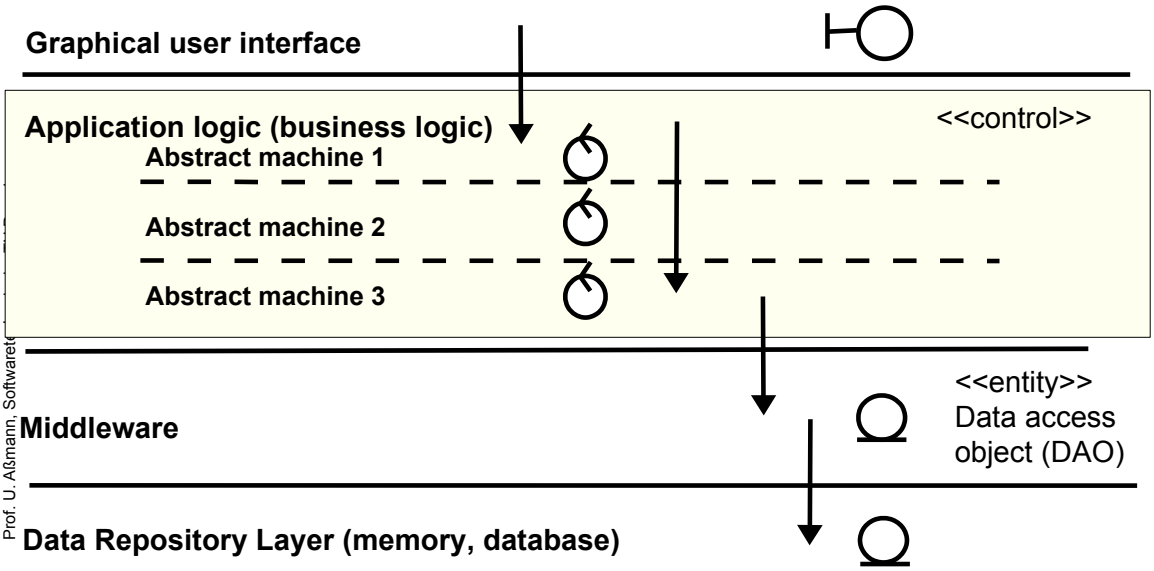
mmands
drop)



stem

Strukturierung der Anwendungslogik in Schichten

- Punktweise Verfeinerung der Anwendungslogik in Schichten



Entwurf mit geschichteten Abstrakten Maschinen

- Nutze BCED-Architekturstil
- Identifiziere abstrakte Maschinen in der Control-Schicht
  - Ordne sie in Schichten an
  - Denke über ihr “Schichtengeheimnis” nach
  - Kapsele eine Schicht hinter eine Fassade (und in ein Paket)
  - Statte die Fassade mit einer Steuerungsmaschine aus
- Vorteile:
  - Einfachheit
  - Hohe Kohäsion, niedrige Kopplung
  - Gute Austauschbarkeit
  - Gute Variierbarkeit
- Gehe für die anderen Schichten ähnlich vor

The End