Model-Driven Software Development 3. – Tool, Automata, Material Methodology (TAM)

Prof. Dr. Uwe Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

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

WS 15/16-1.1, 19.10.15

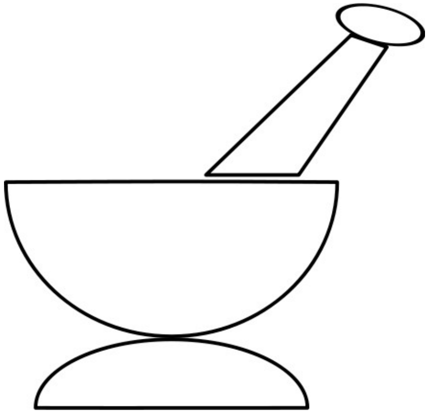
1) Taxonomy of applications, tools and materials
2) TAM for Layering of Applications
3) Basic Functions of Tools
4) Graph-Fact-Isomorphism

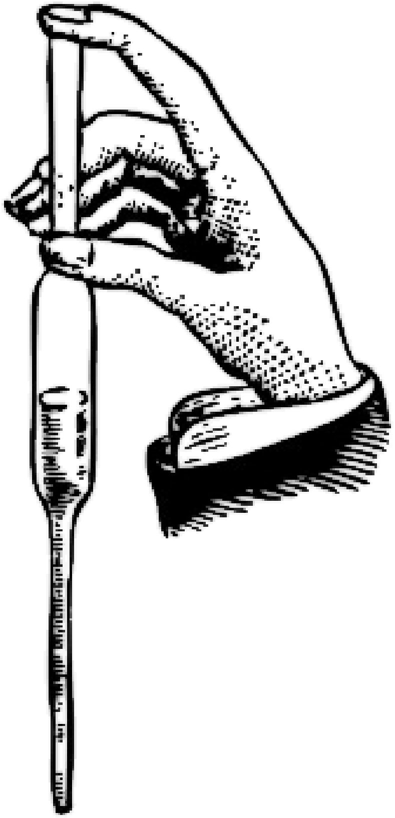
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

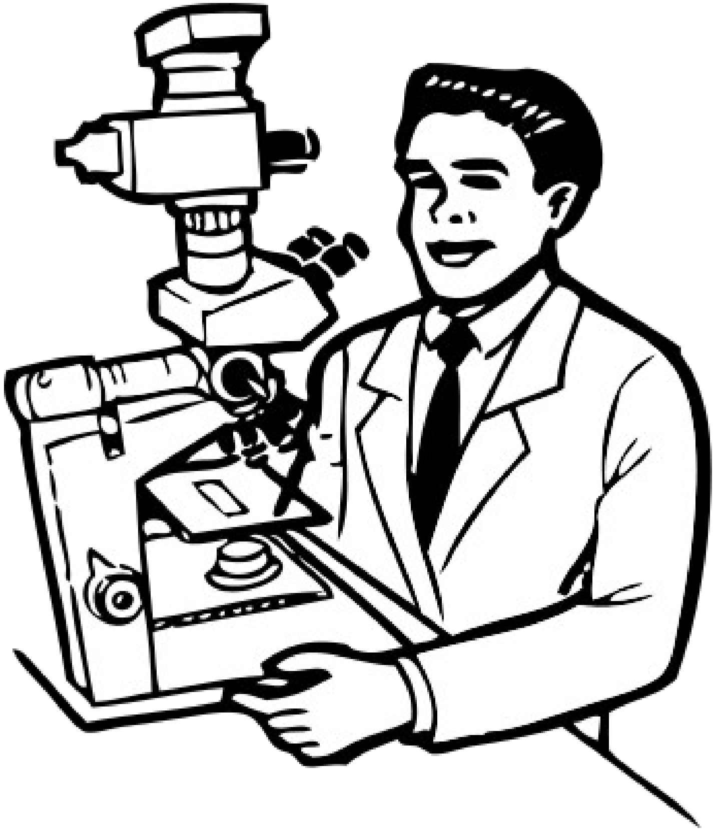
3.1 Tools, Workflows and Materials as Pattern Language for Applications

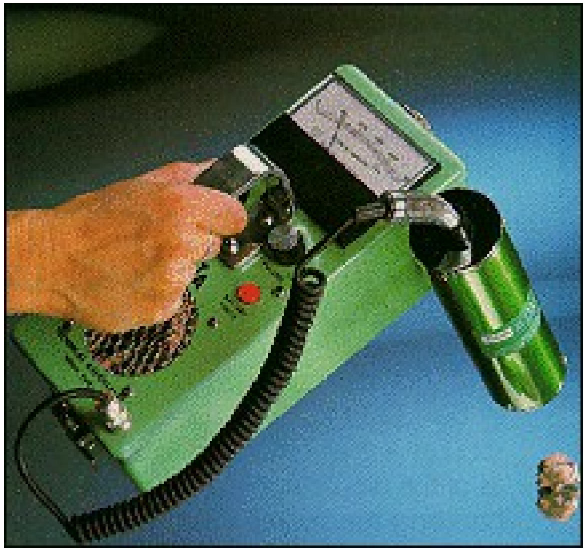
The Tools-Automaton-Material Metaphor

- Any application is built with **tools, automata,** and **materials.**









A Tool or a Material?

- With tears in his eyes the violinist Aaron Rosand left his soul behind in a London hotel suite last week.
- That is how he described the sale of the instrument he had played for more than 50 years, the ex-Kochanski Guarneri del Gesù. The buyer was a Russian billionaire whom Mr. Rosand declined to identify and who paid perhaps the highest price ever for a violin: about $10 million.
- "I just felt as if I left part of my body behind," Mr. Rosand said on Wednesday, overflowing with metaphors for what the instrument meant to him. "It was my voice. It was my career."
- Daniel J. Wakin. New York Times Oct 21, 2009.
  - http://www.nytimes.com/2009/10/22/arts/music/22violin.html?_r=0

https://en.wikipedia.org/wiki/Aaron_Rosand

Human Beings Use Tools

A **Tool (Werkzeug)** is a thing helping to do actions faster as by hand.
An **IT-tool** is a tool running on a computer.
A **data tool** is an IT-tool working with data.

A **software tool** is an IT-tool working on software.
A **modeling tool** is a software tool working on models.
An **application** contains several data or software tools.

A **machine tool (Werkzeugmaschine)** is a tool for production of other tools.

A **software machine tool (Software-Werkzeugmaschine)** is a software tool for production of other software-tools.

- SW-machine tools are the basis of all productivity and wealth

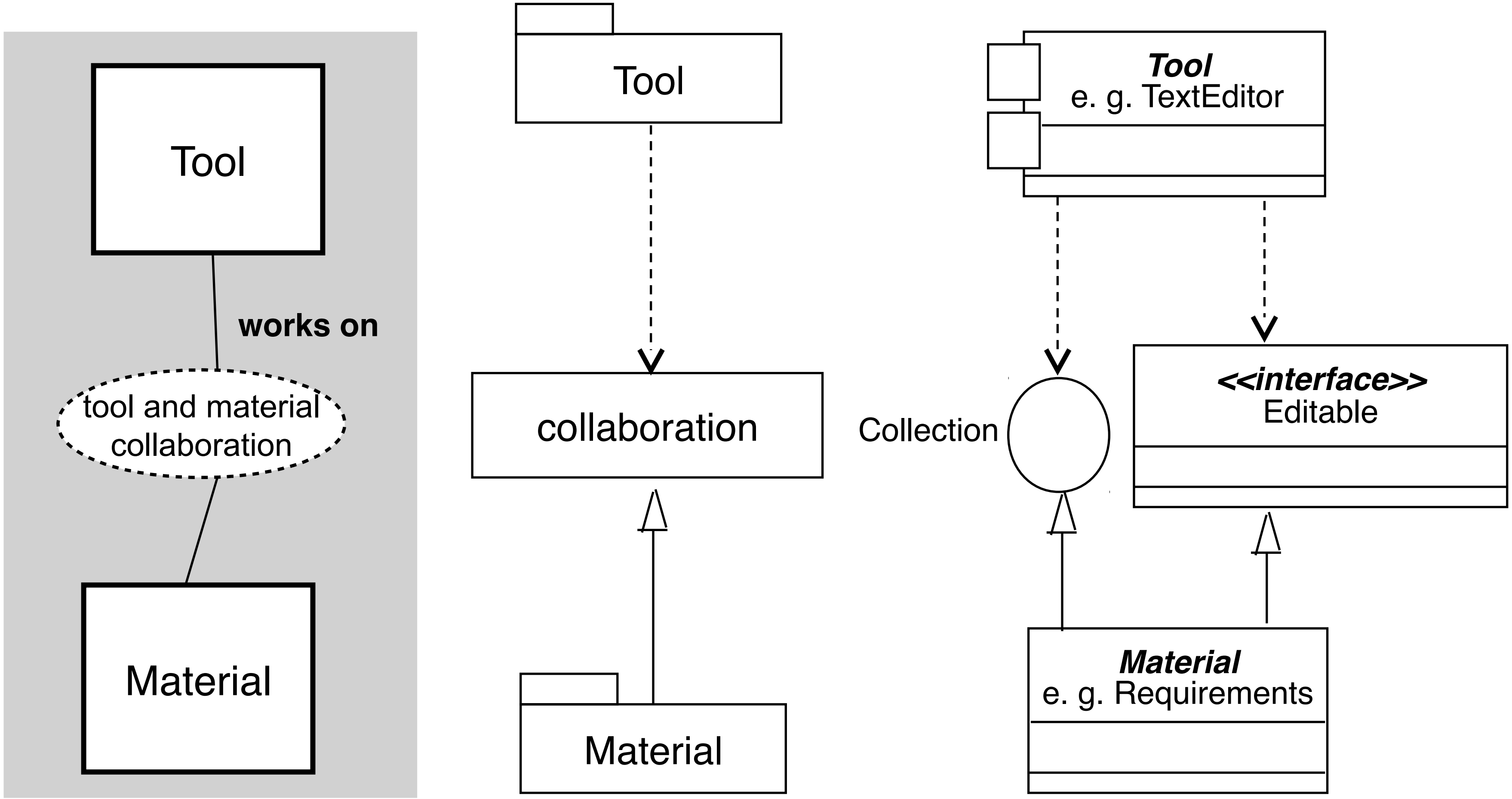
“Tools and Material”-Metapher (TAM) for Programming Applications

- **Tool**: A **tool(-object)** is an active software object that can be used to change material
    - Tools can be used by humans (interactively, batch) or by other tools, or by automata (workflows)
- **Material**: A material is a passive object which is handled by a tool
- **Automaton (Workflow engine)**: An Automaton is an operational workflow orchestrating together several tools
- The **collaboration** of Tools und Material is described by a collaboration scheme (role model, Rollenmodell) (see Softwaretechnologie, DPF).

All applications consist of tool-objects in workflows working on material. (Züllighoven principle)

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

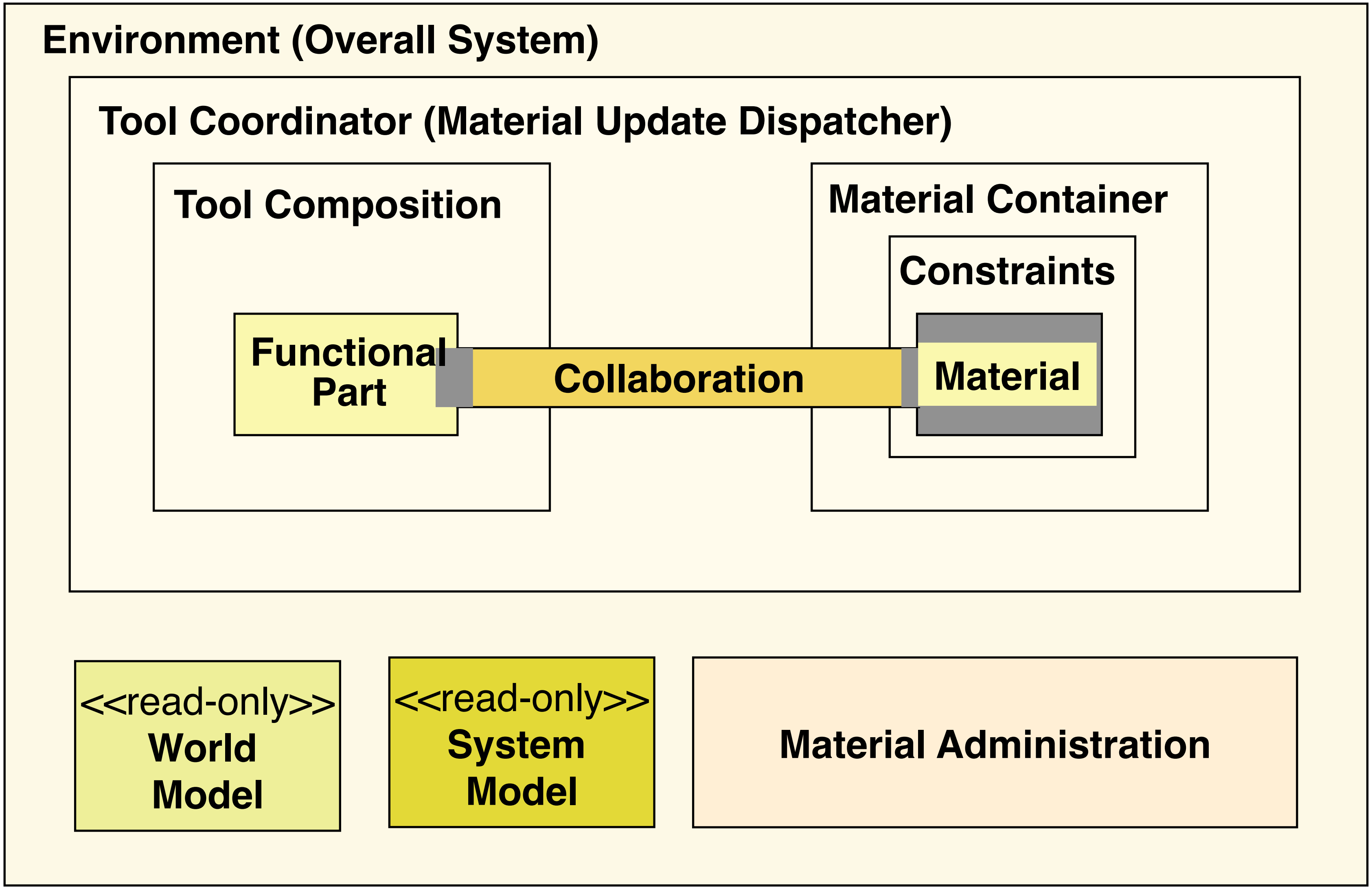
Tool and Material – Metaphor can be Realized in Many Designs of Tools



Conceptual Pattern Design Pattern Construction

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

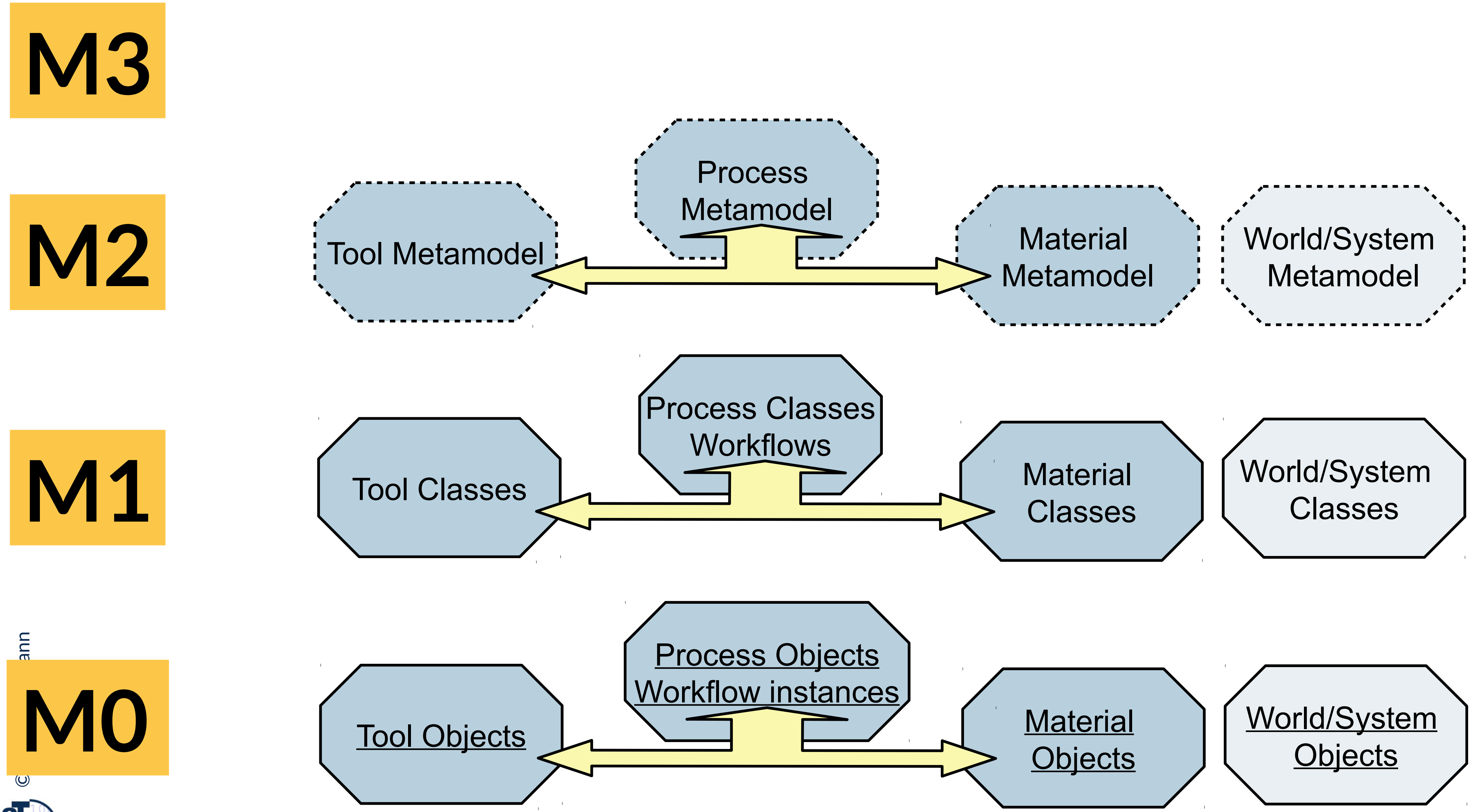
Full TAM Pattern Language Suggests an Architecture for Application Integration


Environment (Overall System)
Tool Coordinator (Material Update Dispatcher)
Tool Composition
Functional Part
Collaboration
Material Container
Constraints
Material
<<read-only>> World Model
<<read-only>> System Model
Material Administration

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

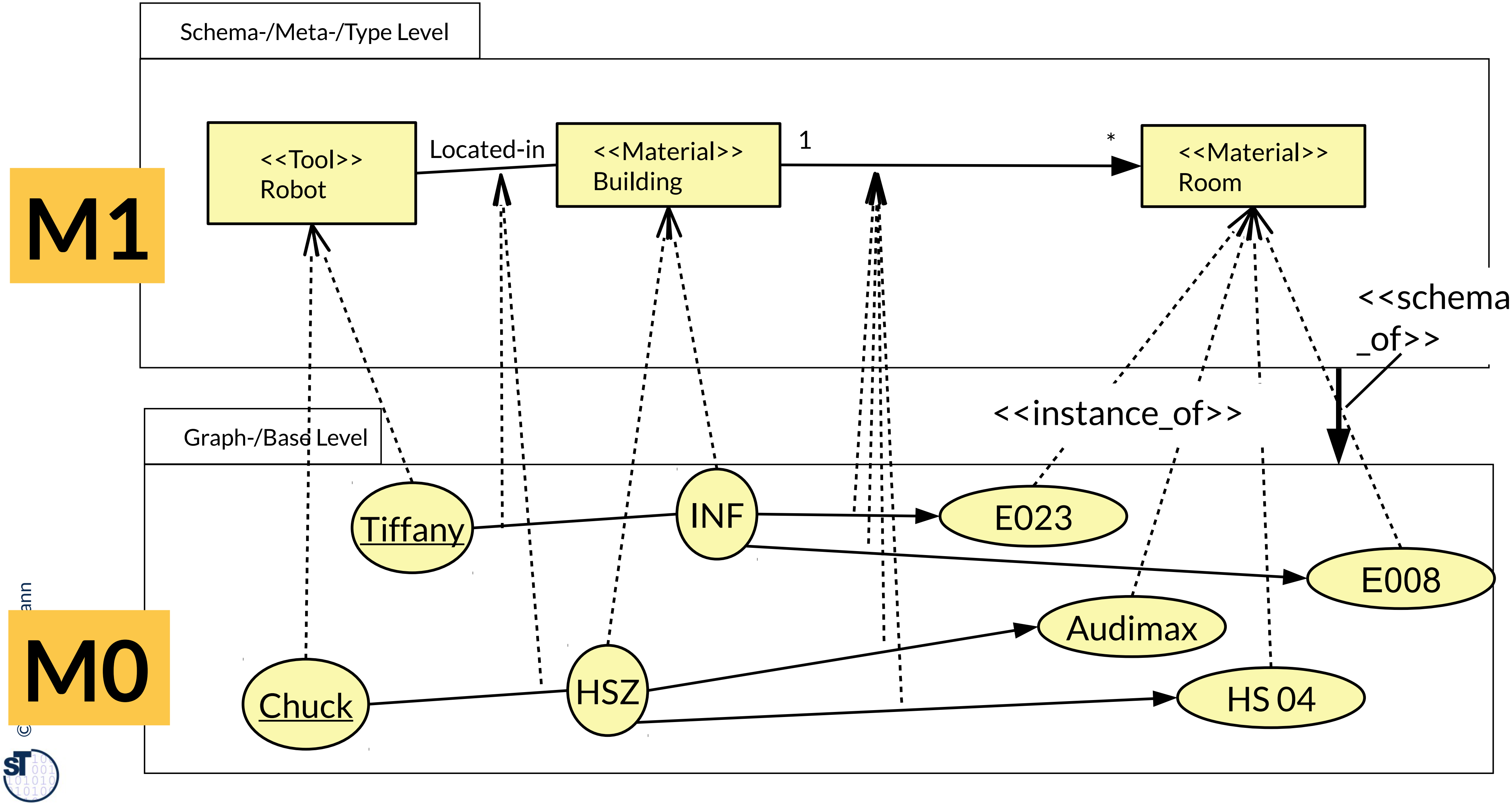
TAM in the Metapyramid

- TAM is a pattern language to structure M0, M1, M2

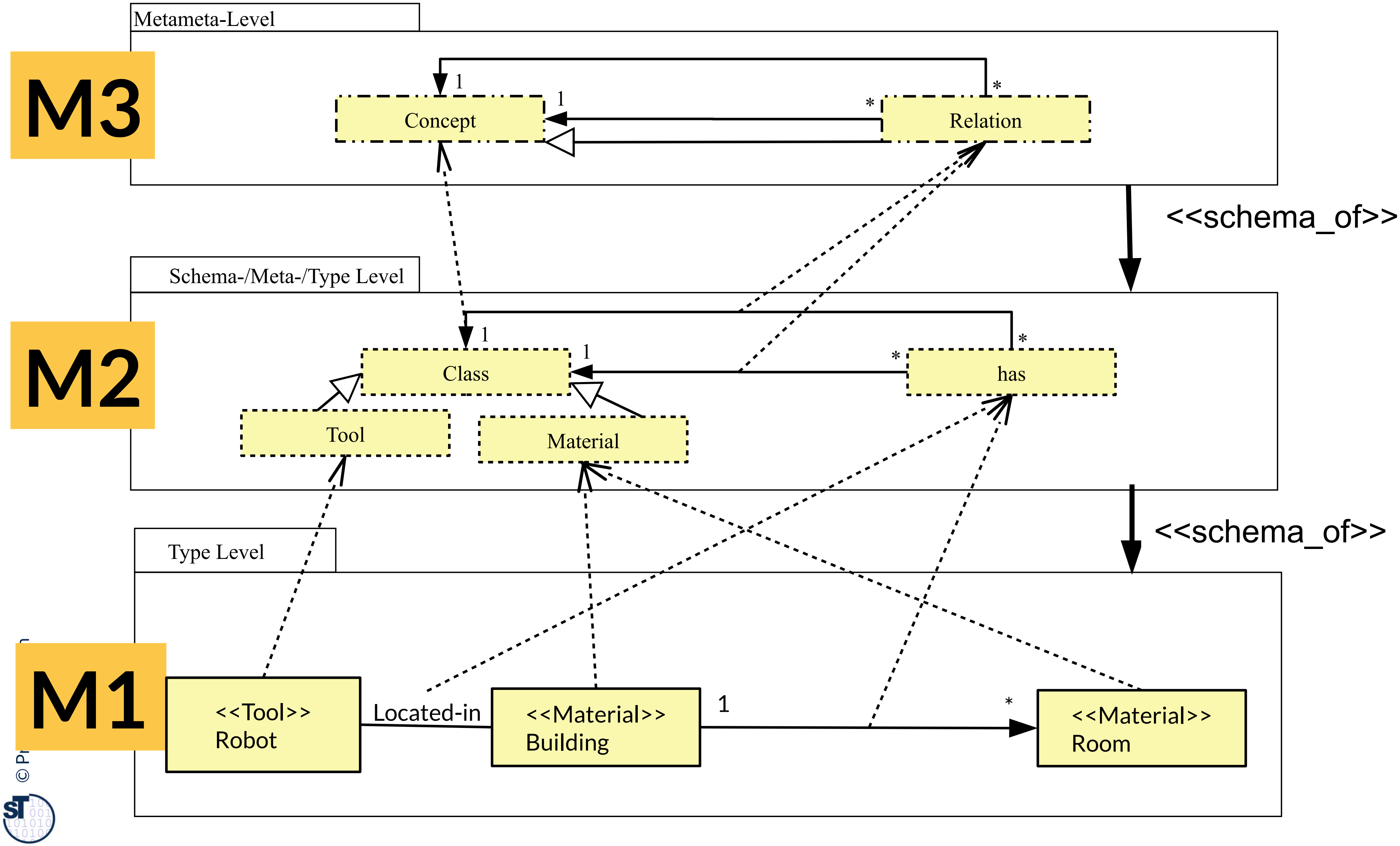


Type Modeling for Application Types (with TAM Tags)

- On M1, also other sets of the application world can be used as types
- Classes can carry the TAM tags



Objects, their Clabjects in Models and Metamodels and TAM



Integrated Development Environment (IDE)
Software-Entwicklungsumgebungen (SEU)

An **integrated development environment (IDE, Software-Entwicklungsumgebung, SEU)** consists of a **structured set of integrated tools** to support a team in software development.

- An IDE is a complex software machine tool (Software-Werkzeugmaschine) for **Computer aided Software Engineering (CASE)**
- A **MDSD-IDE (Meta-CASE)** is an IDE for model-driven software development supporting
  - Many languages (DSL, metamodels) in a technical space
  - Heterogeneous software development
  - Model management system
  - Macromodel
- Other terms
  - Integrated Computer Aided Software Engineering (I-CASE)
  - 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

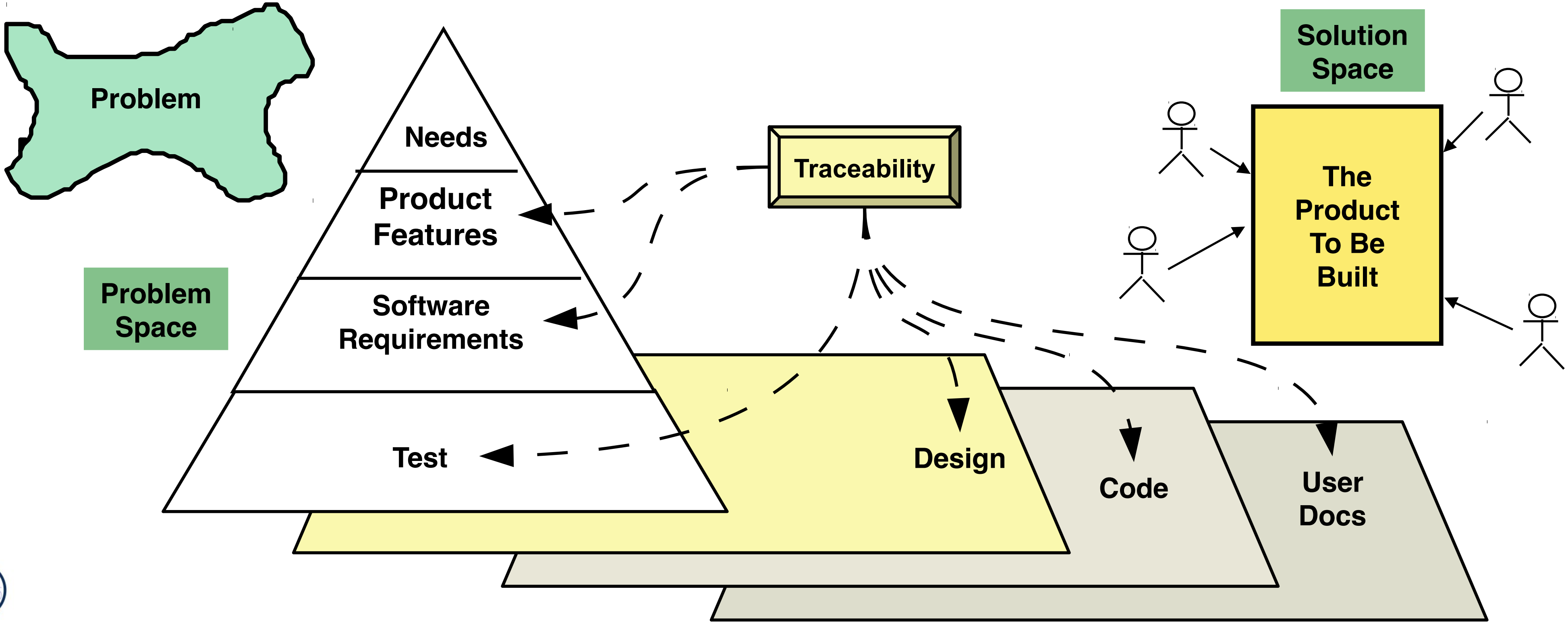
MDSD Applications

An **Model-driven application** consists of a **structured set of integrated tools working** on a **integrated set of materials (typed models),** possibly **in a world model.**

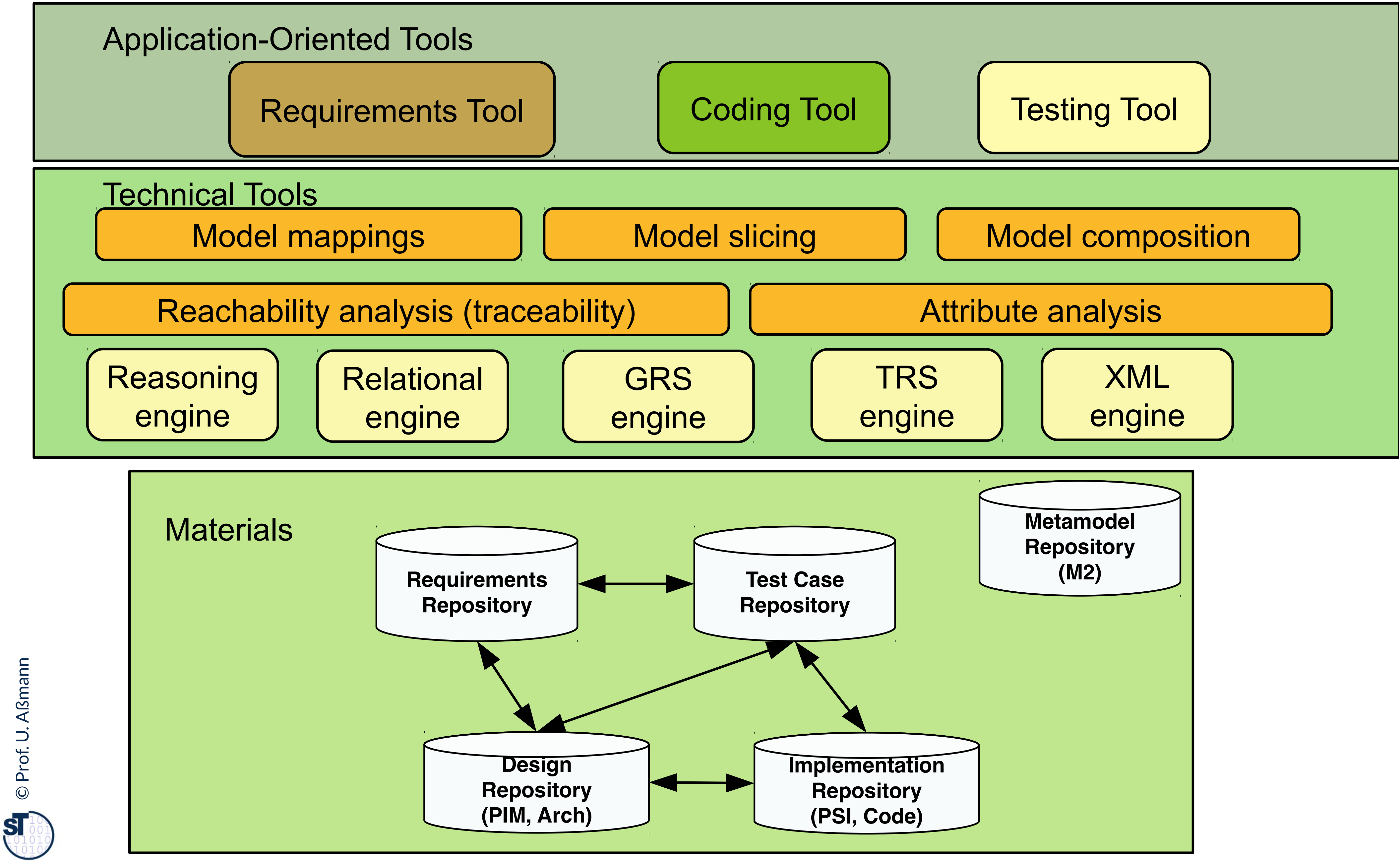
- An MDSD application is also structured with TAM, but uses heterogeneous models.

Q1: IDE and 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
- MDSD relies on model mappings between requirements, test cases, design, and code
- IDE provide tools for all singular aspects, as well as model mappings



Q2: Tool-Objects and Materials in an Integrated Development Environment (IDE, SEU) for MDSD



Application-Oriented Tools

Requirements Tool

Coding Tool

Testing Tool

Technical Tools

Model mappings

Model slicing

Model composition

Reachability analysis (traceability)

Attribute analysis

Reasoning engine

Relational engine

GRS engine

TRS engine

XML engine

Materials

Requirements Repository

Test Case Repository

Metamodel Repository (M2)

Design Repository (PIM, Arch)

Implementation Repository (PSI, Code)

3.3 Identification of Tools, Materials for Layering of Applications

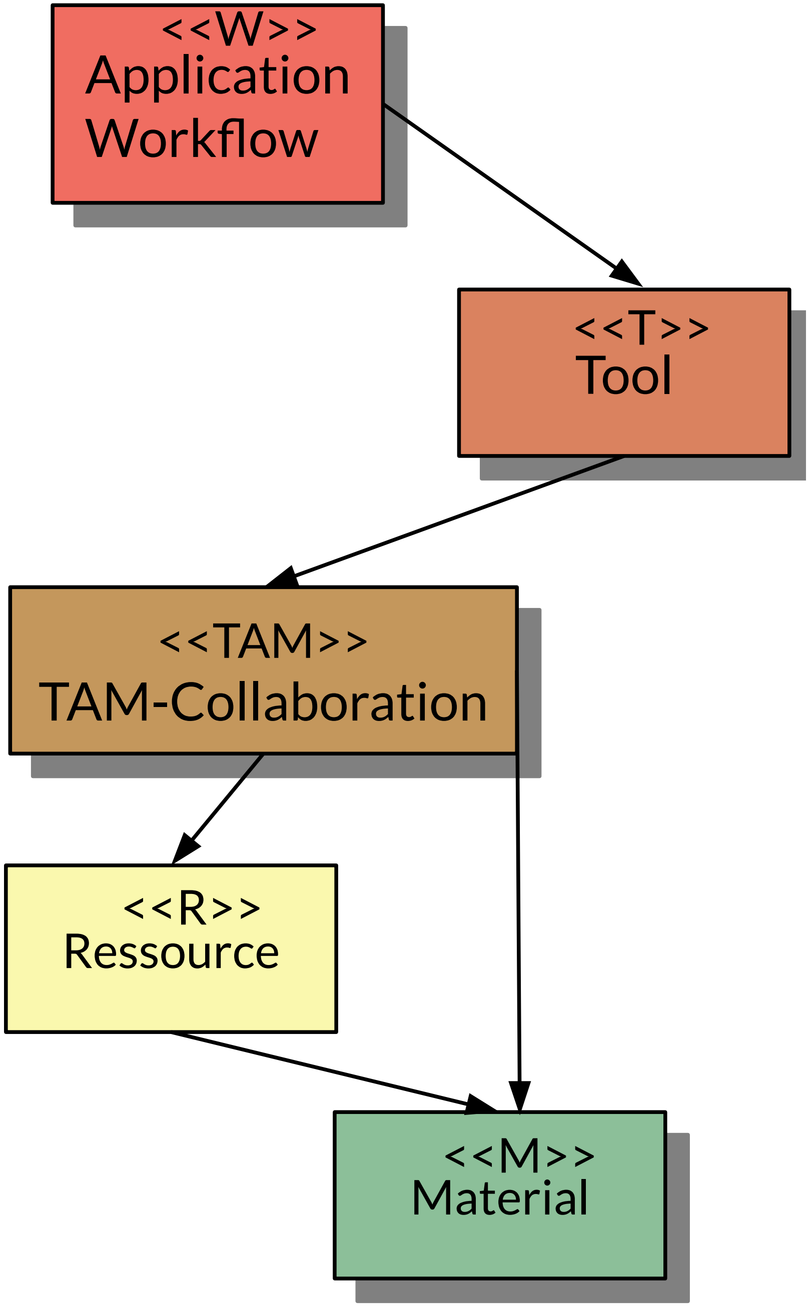
Special kinds of tools, workflows, materials

Perspektive Model TAM: Separation of active and passive Components

Tools-and-Materials [Züllighoven] is a perspektive model with the following aspects:

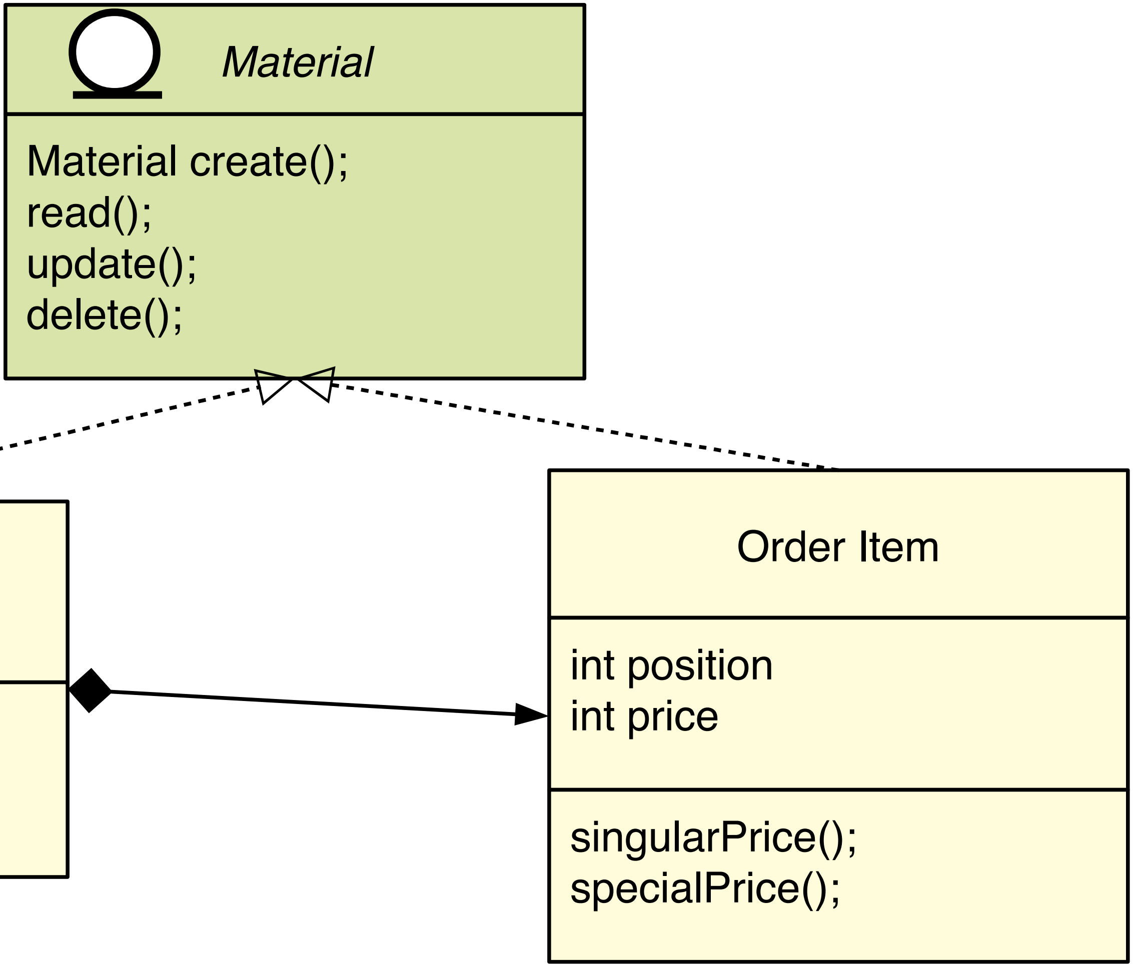
1) Tools (active processes)
2) Ressources (allocatable)
3) Materials (passive data)
4) TAM-Collaboration
5) Workflows (Automata) coordinate Tools

- All program units, such as classes, modules, components, packages can be attributed with these aspects **as stereotypes**

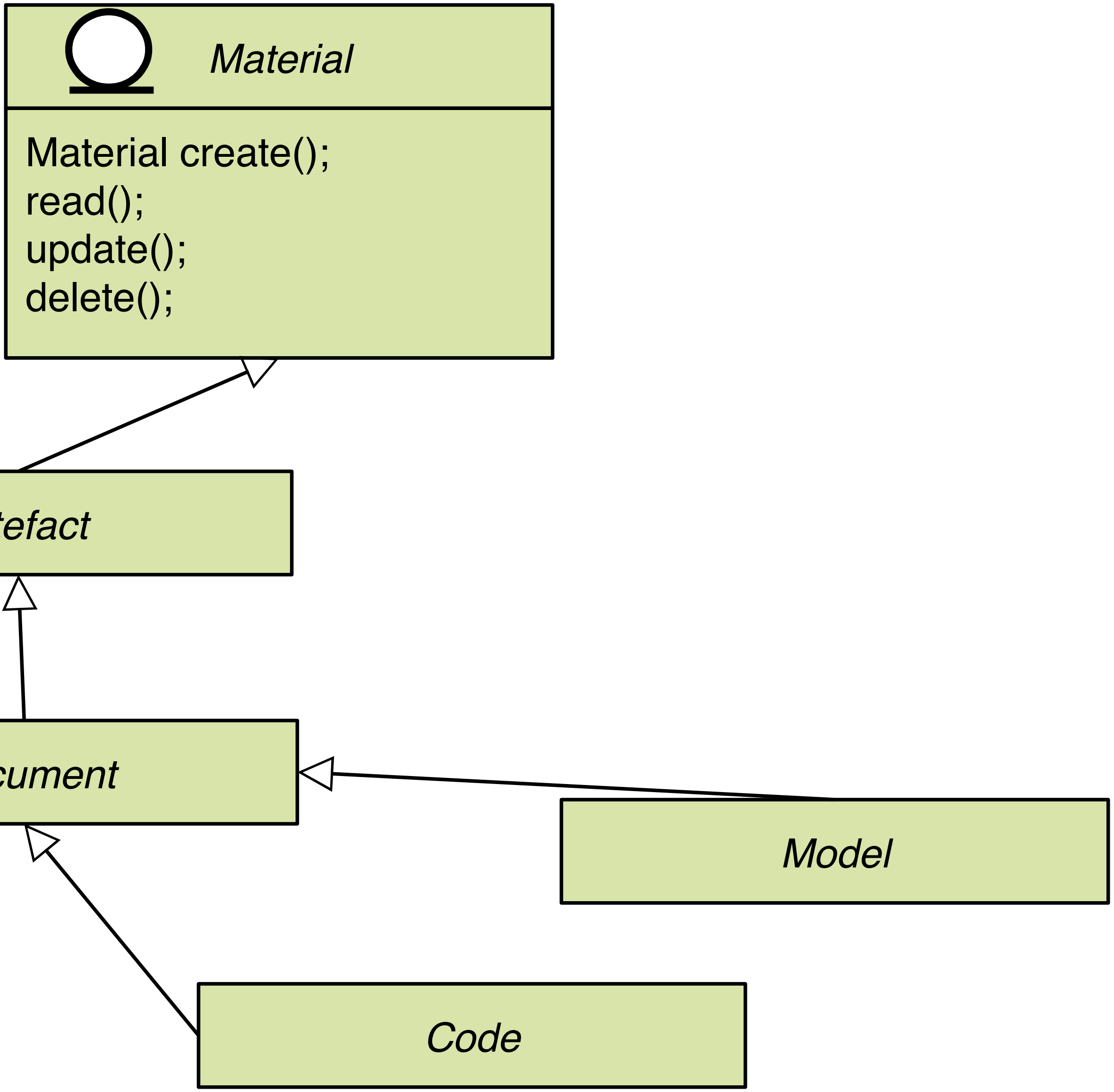


Material-Classes and Interfaces

- Material objects (M0) are passive, e.g., are called from outside
- Material objects can be composite (Pattern Composite or Bureacracy)
- Materials have a CRUD-interface

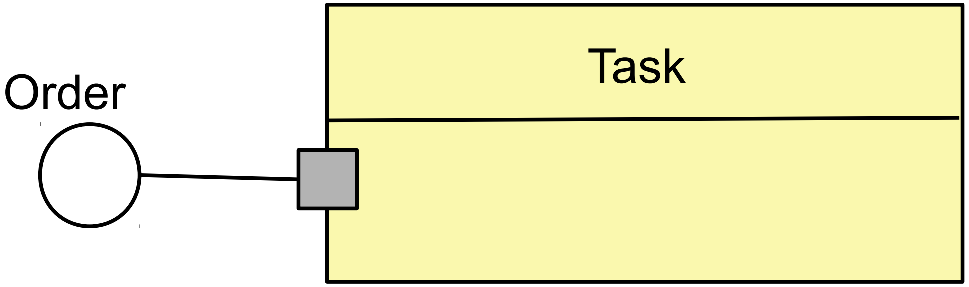


The Material Hierarchy



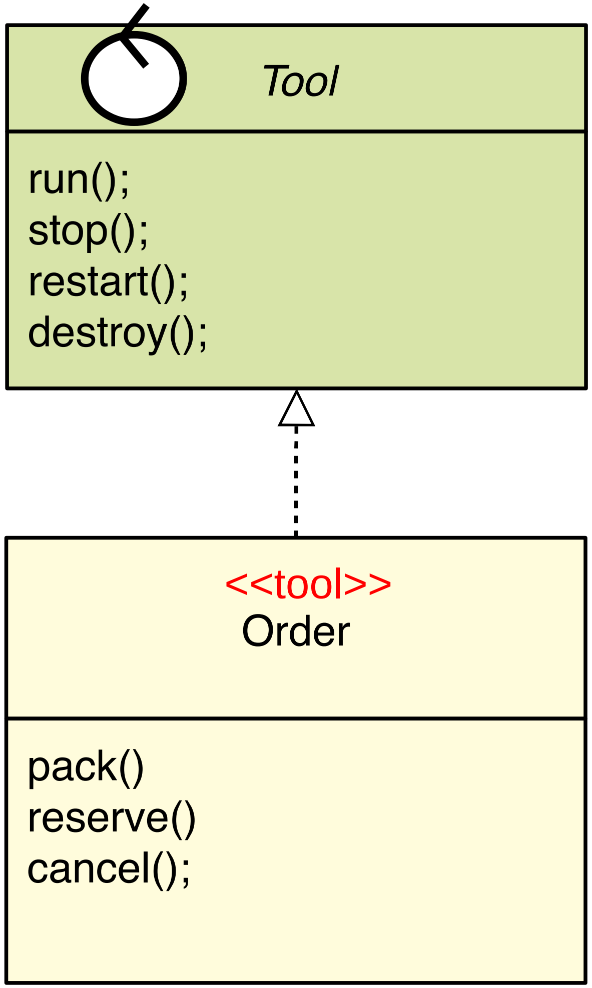
Material-Classes and Interfaces

- Material Classes can appear as interfaces in Ports of UML-components



Tool-Classes and Interfaces

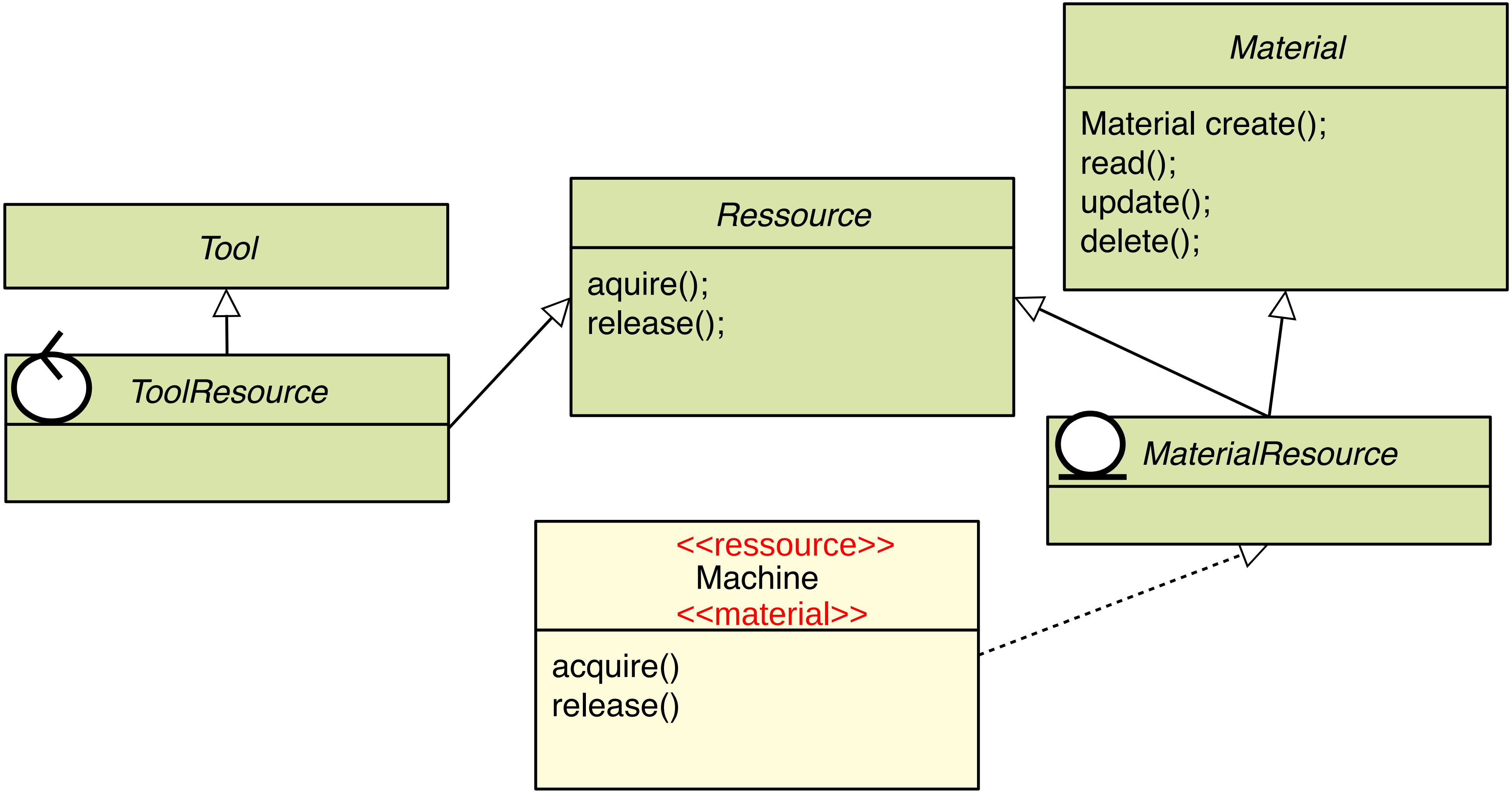
- Tool-objects are active, and have their own thread of control (process)



[Züllighoven]

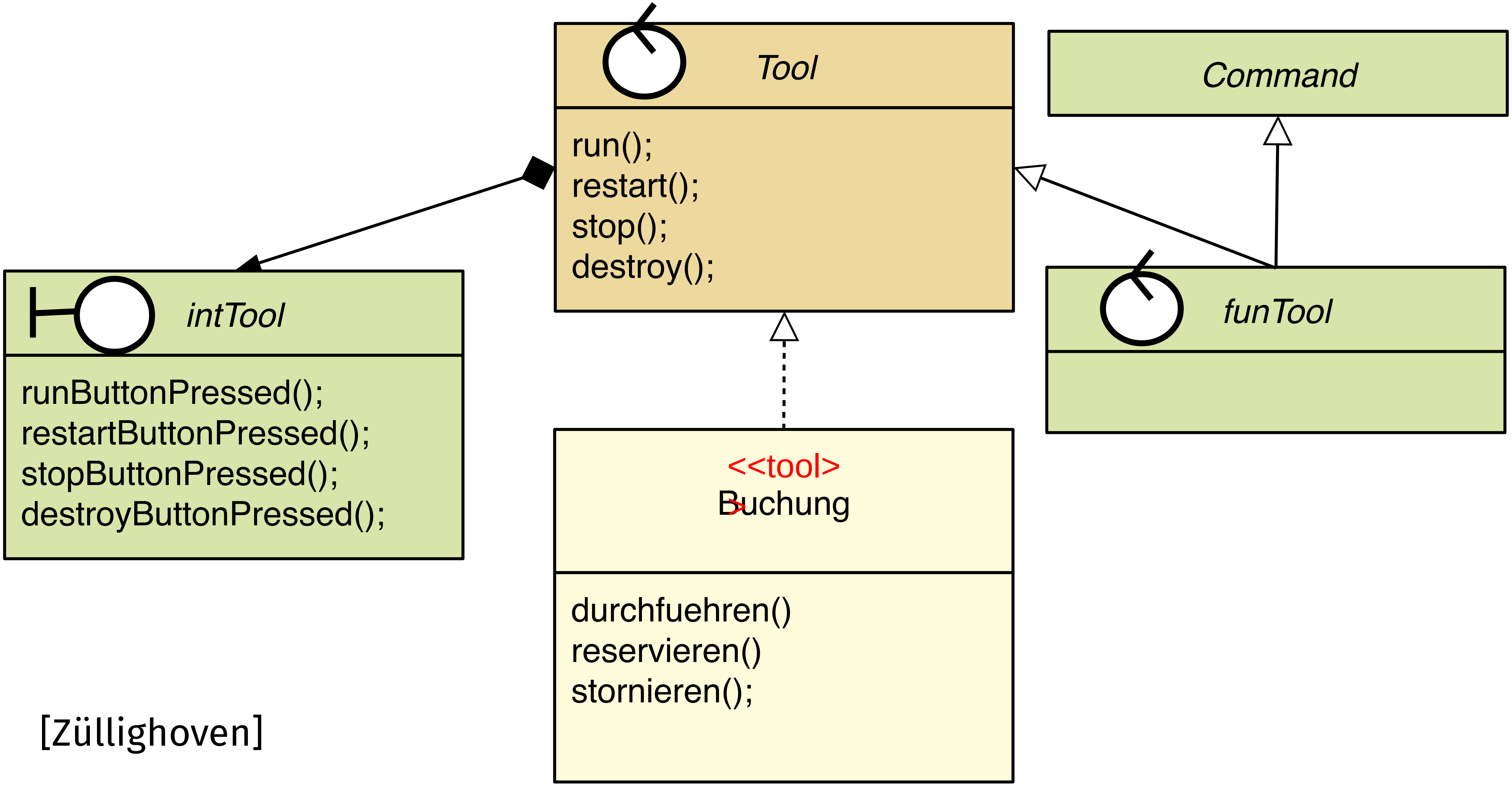
Resource-Interfaces

- **Resource objects** are Tool-Objects or Materials, which must be *allocated* before use and *freed* after use
- **Material resources** are passive. **Tool resources** are active



Tool-Classes and Interfaces

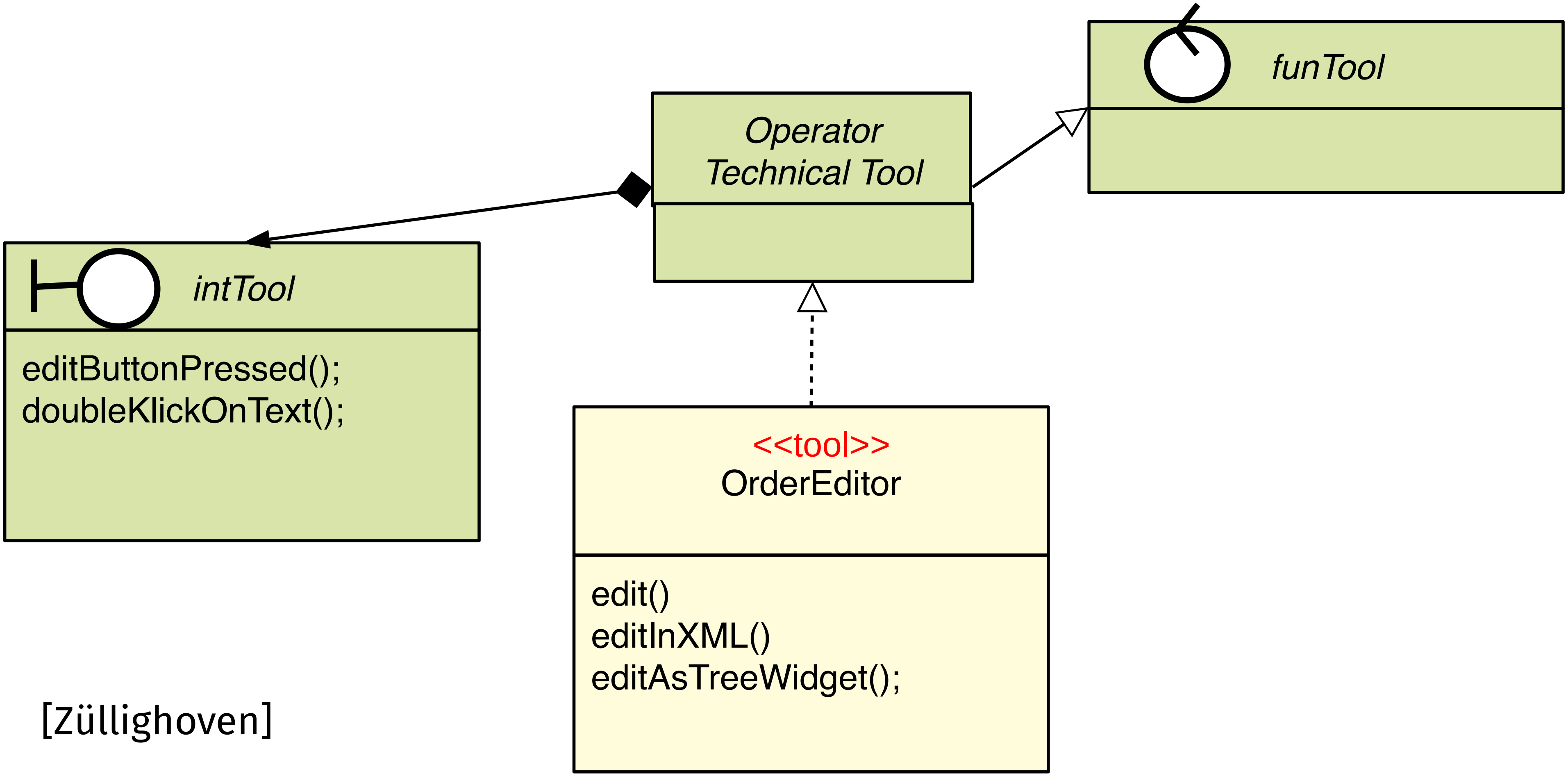
- Tool-objects have an interactive Teil (intTool, boundary) und einen ausführenden, funktionalen Teil (funTool, control), der aus dem Command-Pattern abgeleitet ist
- **Interaktive Tools** stecken hinter den Menüeinträgen



[Züllighoven]

Operator-Classes and Interfaces

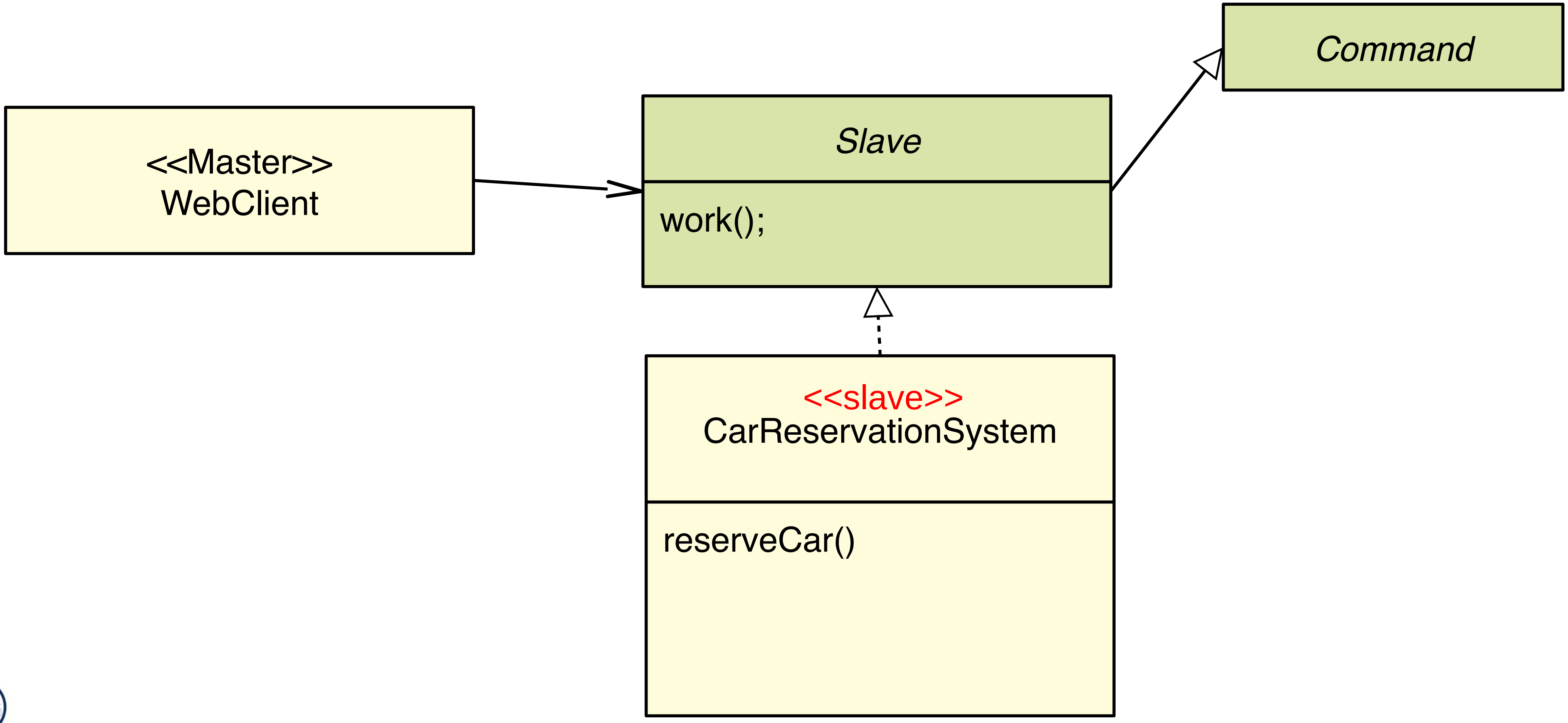
- **Operators (Technical Tools)** on materials carry a technical functionality, which is not specific to an application
    - Bsp.: Editor, Lister, Inspector, Browser, Encryptor, Compressor, Optimizer
- Operators are directly associated with Material
    - They may be part of an algebra on materials



[Züllighoven]

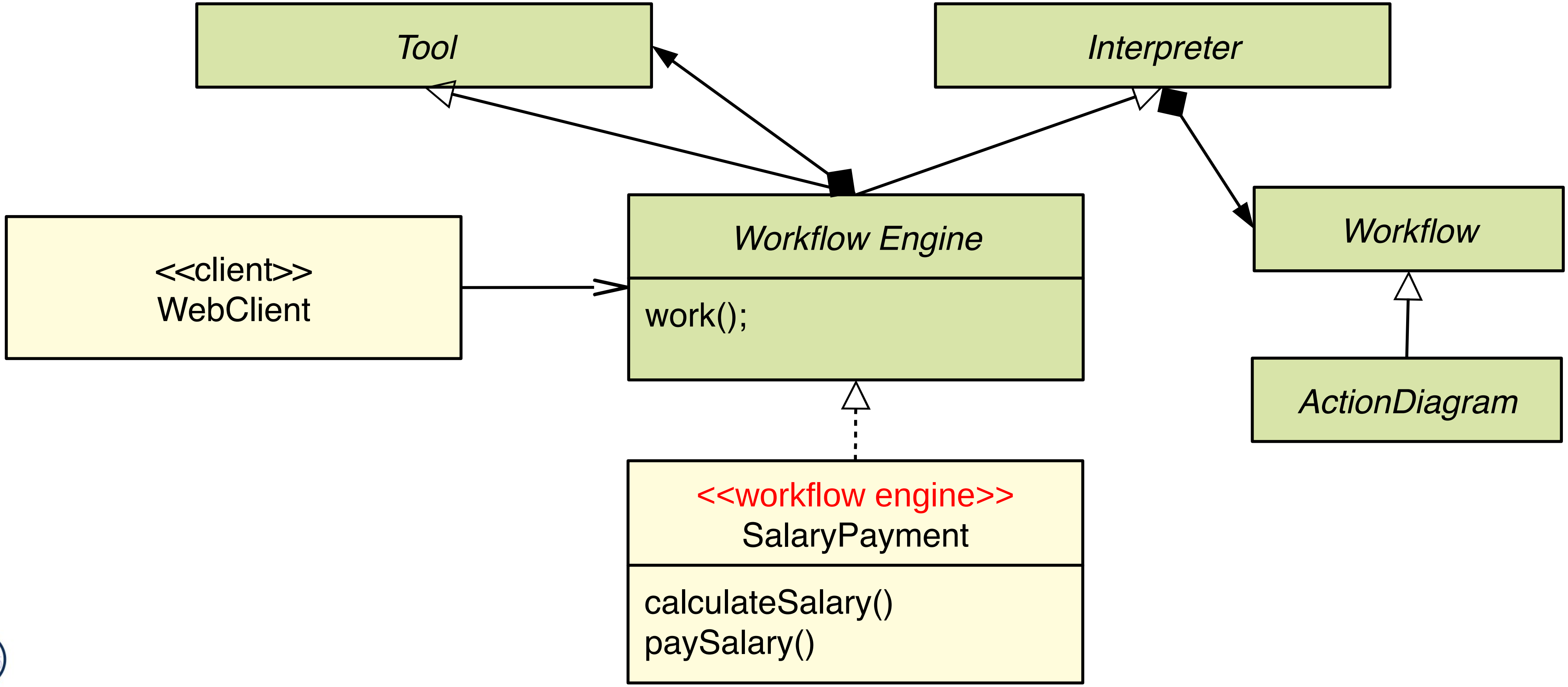
Slave-Classes and Interfaces

- **Slave-Objects** are very specific tools. They are passive, run in batch mode, and return control (Design pattern “Master-Slave”)



Workflow-Engine-Classes and Interfaces

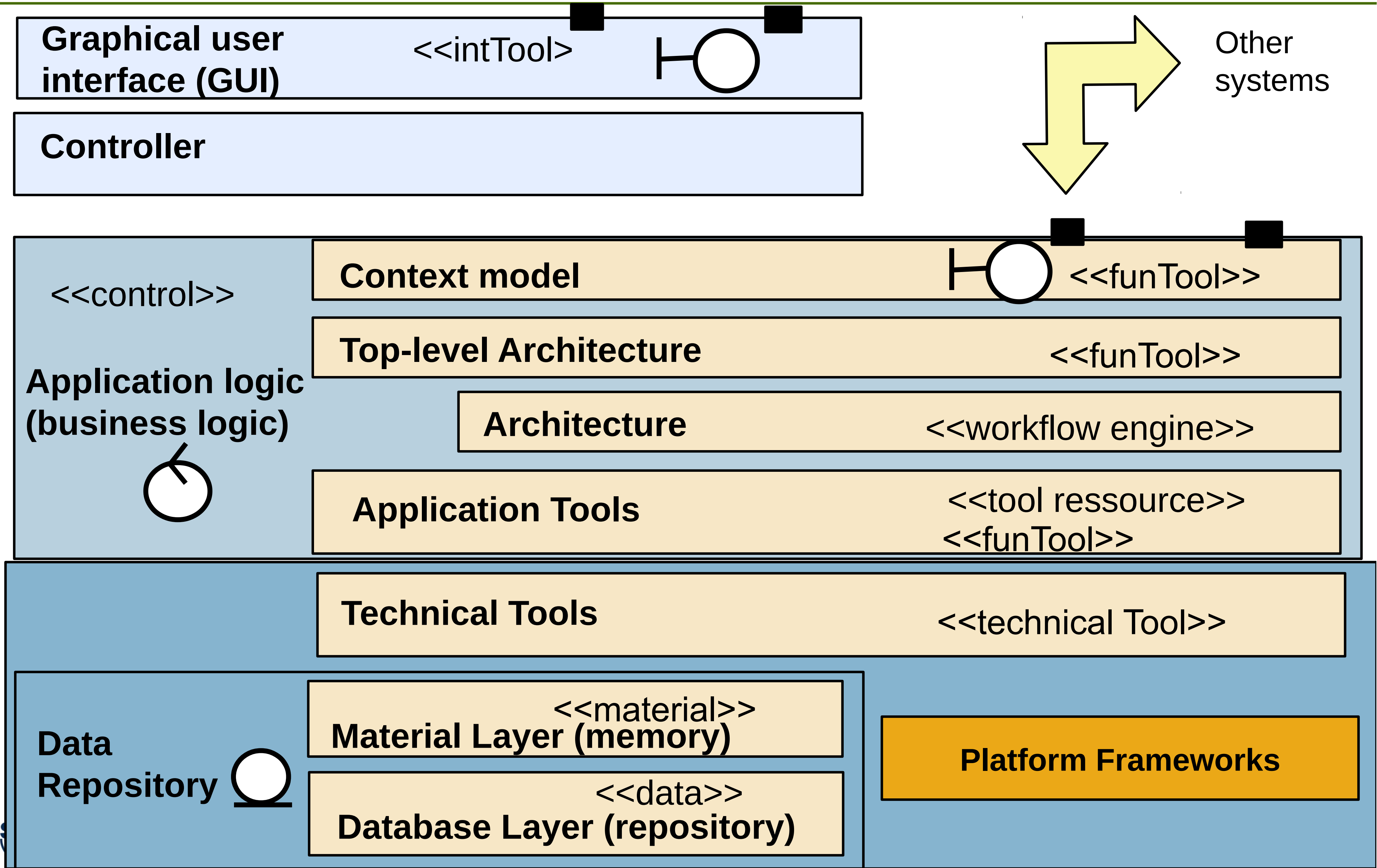
- **Workflow-Engines** are special tools, automata objects organizing a workflow.
  - Workflow-engines interpret the workflow
- Workflow-Engines call other tools
- Their workflows are specified by a behavioral language (action diagrams, statechart, BPMN)



M0 Layers and TAM-Classification

- Die TAM-classification enables to position objects in the layer cake of the application (M0 layer cake)

Q3: M0-Layer Cake



Graphical user interface (GUI)

<<intTool>>

Other systems

Controller

<<control>>

Application logic (business logic)

Context model

<<funTool>>

Top-level Architecture

<<funTool>>

Architecture

<<workflow engine>>

Application Tools

<<tool ressource>>
<<funTool>>

Technical Tools

<<technical Tool>>

Data Repository

<<material>>
Material Layer (memory)

<<data>>
Database Layer (repository)

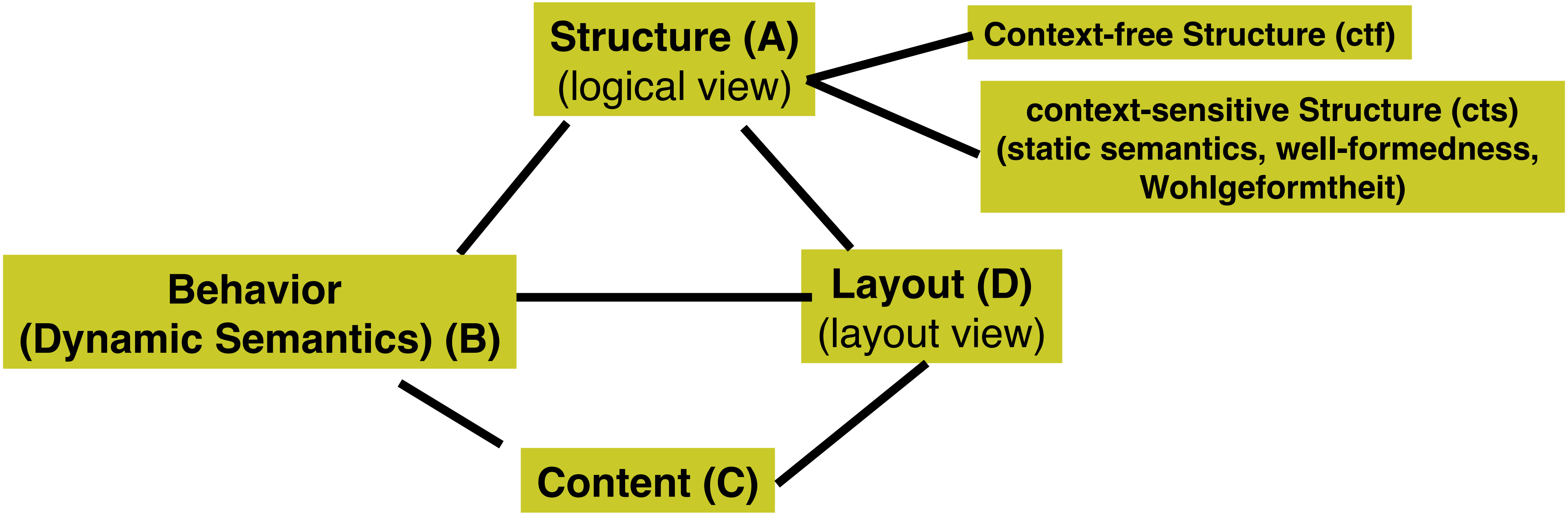
Platform Frameworks

3.3 Basic Functions of Software Tools

Tools on Different Kinds of Materials (Artefacts)

- **Code-centered tools:**
  - **Software** are programs with documentation and test architecture
- **Document-centered tools**
  - Are needed for software
- **Model-centered Tools**
  - Basic for MDSD IDE

Aspects of Materials (Documents, Models, Code)



- **Structure**: log. Units
  - Context-free: Hierarchic structure
  - Links: cross links, references
  - context-sensitive structure mit consistency conditions for well-formedness (static semantics)
- **Semantics**: Programme besitzen eine Bedeutung (dynamische Semantik, Verhalten)
- **Content**: Text, Grafics, images, videos
- **Layout**: Placement

Well-Formedness of Materials (Models, Documents, Code)

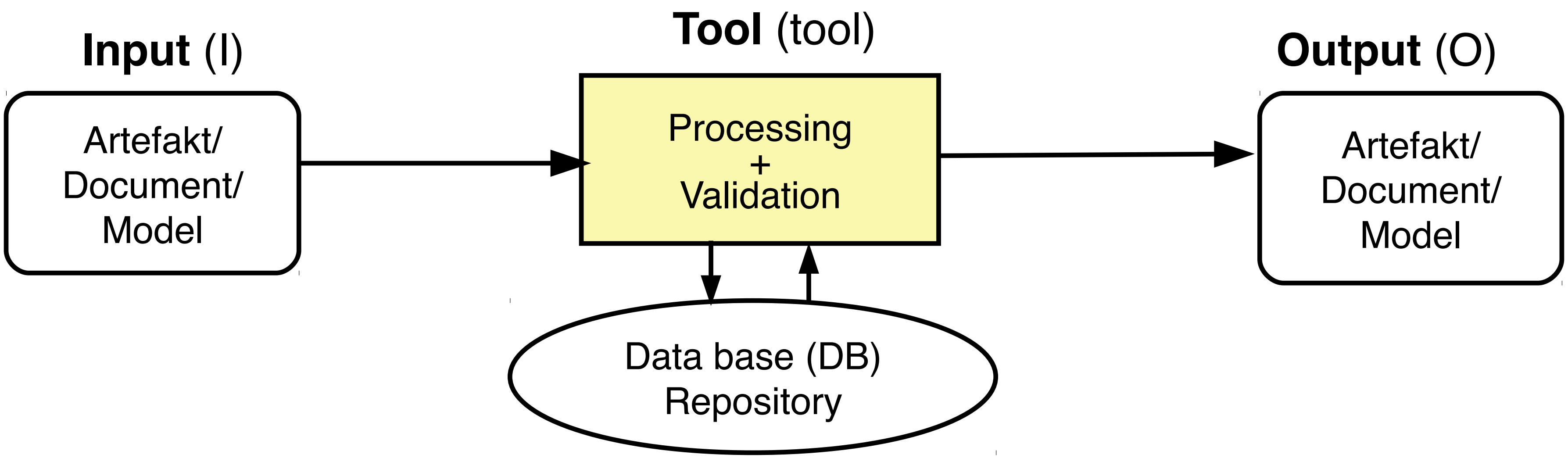
An artefact is **well-formed**, if it fulfils context-sensitive constraints (integrity rules, consistency rules).

Tools check consistency rules on materials by **semantic analysis (context analysis of material constraints)** in the **material container**:

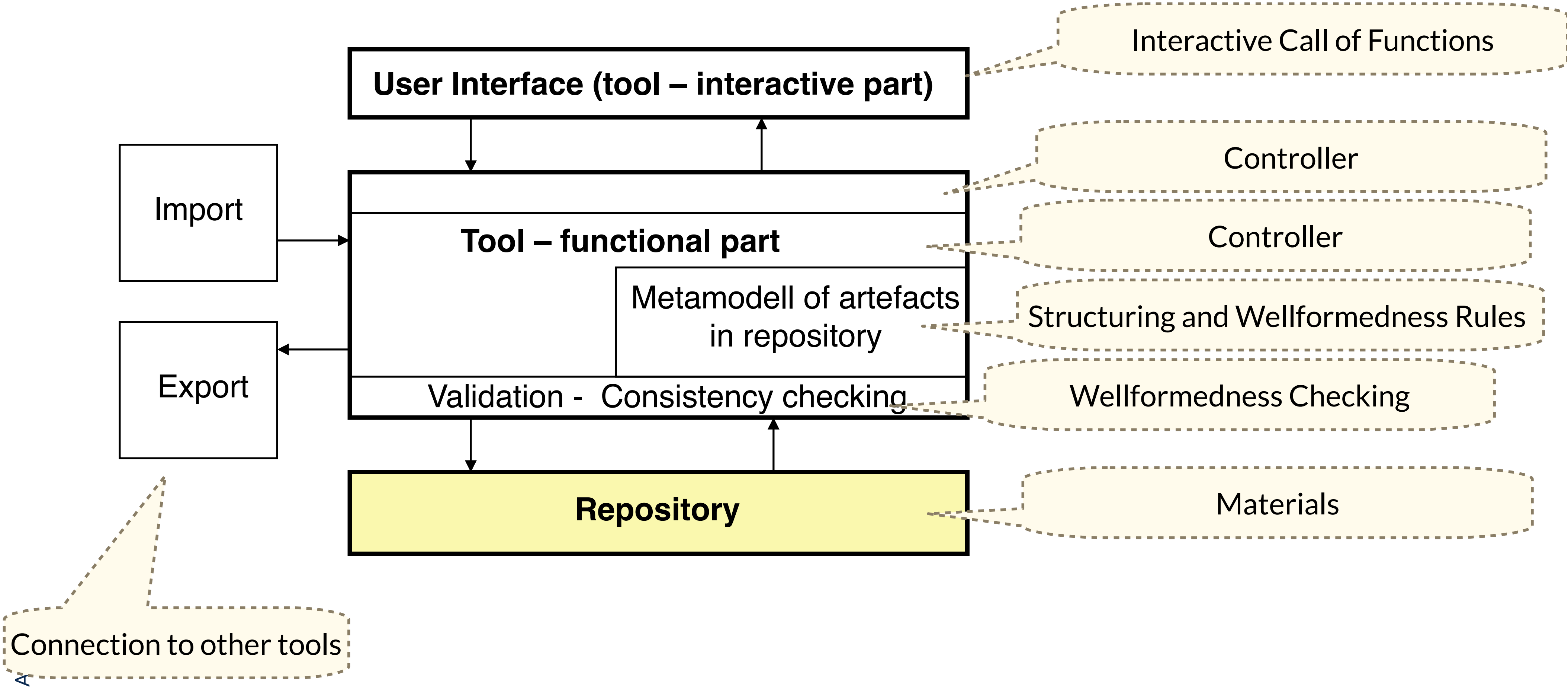
- Layout rules forbid loose or ugly layouts
- Name analysis finds the meaning of names
- Links are set correctly
- Range checks (Bereichsprüfungen) check validity of ranges of values
- Structuring of data structures (see ST-II)
  - Azyclicity, layering, Reducibility
  - Strongly connected components
- Vorbidden combinations

Tools are Deterministic Functions

$\boxed{\text{tool: I x DB} \rightarrow \text{DB x O}}$


Input (I)
Artefakt/ Document/ Model
Tool (tool)
Processing + Validation
Output (O)
Artefakt/ Document/ Model
Data base (DB) Repository

Q4: Logic View of Tool Architecture


User Interface (tool – interactive part)
Import
Export
Tool – functional part
Metamodell of artefacts in repository
Validation - Consistency checking
Repository
Interactive Call of Functions
Controller
Controller
Structuring and Wellformedness Rules
Wellformedness Checking
Materials
Connection to other tools

After: [Bal-II, S. 604]

Artefact Types

- Free text
  - Word documents, requirement specifications, user stories, comments
- Models
  - Textual models
    - Trees and ordered trees (terms)
      - S-Expressions (Lisp, Scheme)
      - Link trees (XML-trees, JSON-trees)
      - Feature terms
    - Ontologies
  - Diagrammatic models, usually specific graphs
    - Analysis documents and design specifications (UML-diagrams), Petri-Nets, statecharts
- Graphics: Visualizations in 2-D or 3-D
- Tables: Relations, test case tables
- Code: e.g., Pseudocode, code templates, source code

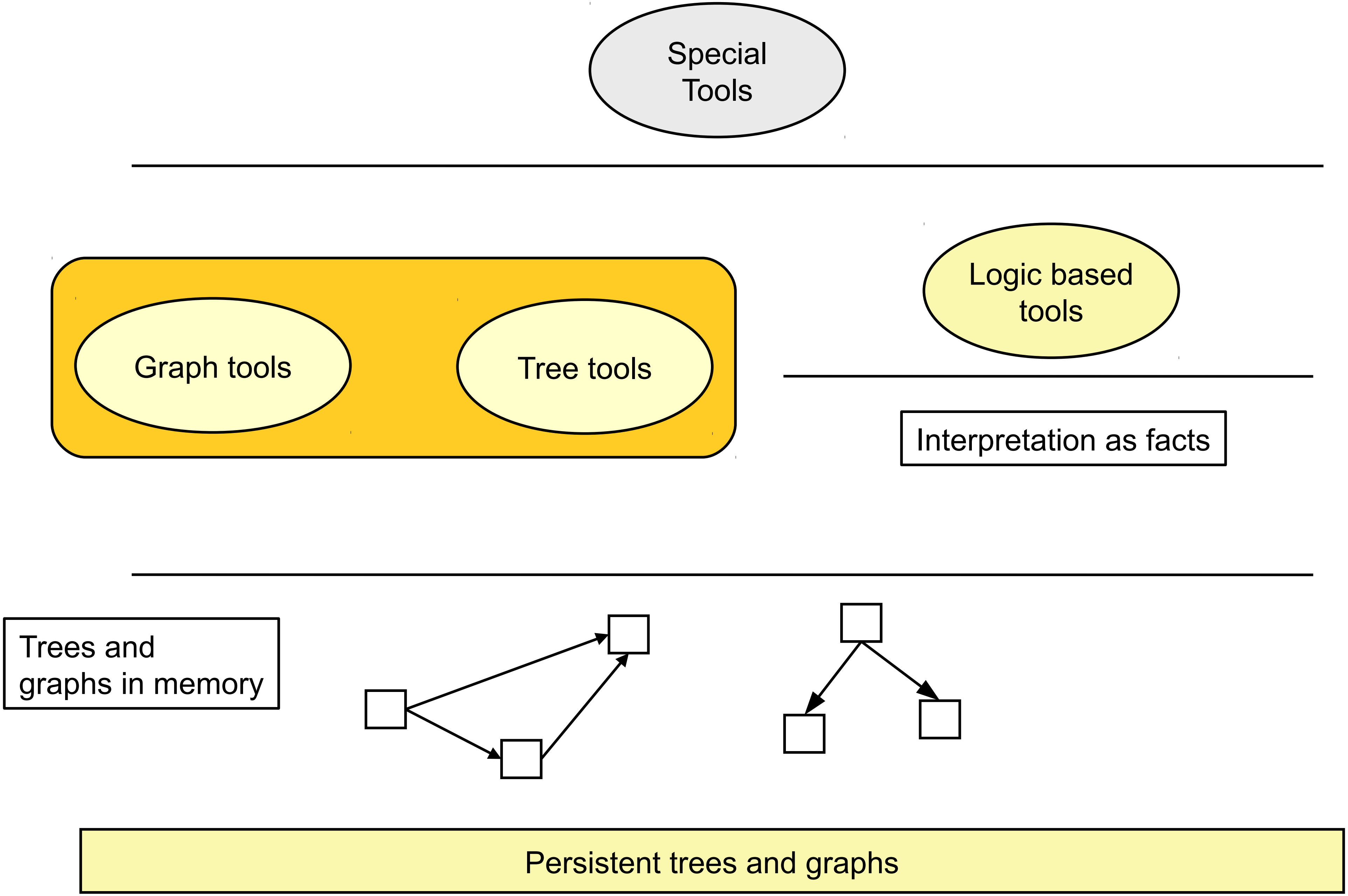
after: Denert, E.: Dokumentenorientierte Software-Entwicklung; Informatik-Spektrum 16(1993) H. 3, S. 159 - 164

3.3.2 The Graph-Fact-Isomorphism

The Graph-Fact-Isomorphism

- Every Graph can be represented as a fact base of a logic inference engine (reasoner)
- Every fact base can be interpreted as Graph
  - binary: Graph
  - n-ary: Hypergraph
- Therefore, logic inferencers and graph transformation tools can be used on the same data and artefacts
- Materials can be seen as facts of a reasoner or graphs of a modeling environment
- *Metamodeling* uses both kinds of technologies

IDE with Logic-based and Graph-based Tools



Special Tools

Graph tools

Tree tools

Logic based tools

Interpretation as facts

Trees and graphs in memory

Persistent trees and graphs

The End

- Explain the consequences of the Züllighoven principle for the construction of heterogeneous applications
- Why does the TAM pattern language cross the metapyramid?
- Which concepts belong to a process metamodel in contrast to a tool or material metamodel?
- Why is static semantics divided into context-free structure and context-sensitive wellformedness conditions?
- Why is it possible to store a model in a database or an inferencer?