13) The Tools And Materials Architectural Style and Pattern Language (TAM)

Prof. Dr. U. Aßmann

Chair for Software Engineering

Faculty of Informatics

Dresden University of Technology

12-1.0, 12/15/12

1) Tools and Materials - the metaphor

2) Tool construction

3) The environment

   1) Material constraints

4) TAM and layered frameworks

1

Design Patterns and Frameworks, © Prof. Uwe Aßmann

Literature

2

▶ D. Riehle, H. Züllighoven. A Pattern Language for Tool Construction and Integration Based on the Tools&Materials Metaphor. PLOP I, 1995, Addison-Wesley.

▶ JWAM: Still available on Sourceforge http://sourceforge.net/projects/jwamtoolconstr/

- A copy of jwam.org is in the Internet Archive, also literature
- http://web.archive.org/web/20041009212341/www.jwam.org/engl/produkt/e_literature.htm
- Thanks to Moritz Bartl!

Literature

3

▶ D. Riehle, H. Züllighoven. A Pattern Language for Tool Construction and Integration Based on the Tools&Materials Metaphor. PLOP I, 1995, Addison-Wesley.

▶ JWAM: Still available on Sourceforge http://sourceforge.net/projects/jwamtoolconstr/

- A copy of jwam.org is in the Internet Archive, also literature
- http://web.archive.org/web/20041009212341/www.jwam.org/engl/produkt/e_literature.htm
- Thanks to Moritz Bartl!

Secondary Literature

4

▶ Heinz Züllighoven et.al. The object-oriented construction handbook. Morgan Kaufmann Publishers, 2004. The TAM explained in detail.

▶ In German: Heinz Züllighoven et.al. Das objektorientierte Konstruktionshandbuch – nach dem Werkzeug und Material-Ansatz. Dpunkt-Verlag, Heidelberg, 1998.

▶ D. Riehle. Framework Design – A Role Modeling Approach. PhD thesis 13509, ETH Zürich, 2000. Available at http://www.riehle.org.

Exam Questions (Examples)

- ► What are the central metaphors of the Tools-and-Materials architectural style?
- ► Explain tool-material collaboration. Which roles do role models play?
- ► How are tools structured?
- ► How is TAM arranged as a layered framework?

Why Do People Prefer to Use Certain Software Systems?

- ► People should feel that they are competent to do certain tasks
- ► No fixed workflow, but flexible arrangements with tools
  - ▪ Domain office software, interactive software
- ► People should decide on how to organize their work and environment
- ► People want to work incrementally, in piecemeal growth

13.1 Elements of "Tools and Materials"

13.1 The Central T&M Metaphor

- ► Tools and Materials pattern language T&M
  - ▪ Werkzeug und Material (WAM)
  - ▪ Craftsmanship: Craftsmen use tools to work on material
- ► People use tools in their everyday work: Tools are means of work
  - ▪ People use tools to work on material
- ► T&M-collaboration: Tools and materials are in relation
- ► Environment: Craftsmen work in an environment

And 3-Tier Architectures?

- ▶ Another popular architectural style for interactive applications is 3-tier architecture
- ▶ However, the 3-tiers are so coarse-grained that they do not really help for interactive applications
- ▶ T&M is much more detailed

User Interface
_____

Application logic
— — — — — — — —

Middleware
_____

Data Handling
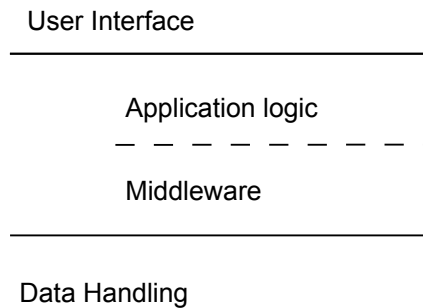
Material

- ▶ Passive entities, either values or objects
  - ▪ Ex.: Forms laid out on a desktop, entries in a database, items in a worklist
- ▶ Prepared and offered for the work to be done
- ▶ Transformed and modified during the work
- ▶ Not directly accessible, only via tools

- ▶ Values (e.g., Dates, Money)
  - ▪ Without time and position
  - ▪ Abstract, without identity
  - ▪ Equality is on value
  - ▪ A value is defined or undefined, but immutable
  - ▪ Cannot be used in a shared way
  - ▪ Structured (then every subvalue has 1 reference), such as documents
  - ▪ are domain-specific, such as business values (value objects

- ▶ Objects (e.g., Persons, technical objects, Bills, Orders)
  - ▪ With time and position
  - ▪ Concrete, with identity
  - ▪ Equality is on *names*
  - ▪ Mutable; identity does not change
  - ▪ Shared by references
  - ▪ Structured (a subvalue may have several references)

Tools

- ▶ Active entitites
  - ▪ Tools are means of work. They embody the experience of how to efficiently work with material
  - ▪ Present a view on the material.
  - ▪ Often visible on the desktop as wizards, active forms,..
  - ▪ Tools give feedback to the user
  - ▪ Tools have a state
- ▶ If well-designed, they are transparent, light-weight, and orthogonal
  - ▪ However, they should not disappear, since users need to look at a tool if they are worried
- ▶ Examples:
  - ▪ Browser – Contents of a folder
  - ▪ Interpreter – Code and data
  - ▪ Calendar - Calendar data
 - ▪ Form editor - Form

---

Tools vs. Material

- ▶ To say, what is a tool and what the material, depends a lot on the concrete task (interpretation freedom)
  - ▪ Pencil -– paper
  - ▪ Pencil sharpener - pencil
- ▶ Tools can be structured
  - ▪ Supertools and subtools, according to tasks and subtasks
  - ▪ e.g., Calendar = AppointmentLister + AppointmentEditor
- ▶ We work with different tools on the same material
- ▶ In implementations, tools are a often realized as a variant of the Command/Objectifier reified actions
  - ▪ They have a function `execute()`

Tools and Materials as Special Role Model

- ► The tool is active, has control
- ► The material is passive and hands out data

```
┌──────────┐   <<use>>   ┌──────────┐
│  Using   │ ─────────►  │  Usable  │
│   Tool   │             │ Material │
└──────────┘             └──────────┘
```
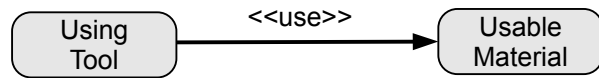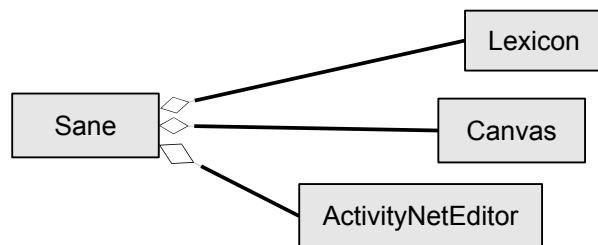
(Work-)Environment

- ► The (Work-)Environment to organize the tools, materials, and T&M-collaborations
  - Tools can be created from the environment by tool factories (Factory pattern)
  - Materials can be created from the environment by material factories
  - Corresponds to the metaphors of a workshop or desktop
- ► Environment for planning, working, arranging, space
  - Several logical dimensions to arrange things

Running Example: TORA Tools

- ► Tool for Task oriented requirements analysis (TORA)
  - Editor SANE for activity nets in requirements analysis
- ► TORA has subtools
  - Glossary browser *Lexicon* to manage glossaries about requirement specifications
  - *Canvas* for the editor's graphical objects. Manipulates the editor's visible materials (Graphical objects, GraphObj):
    - Edit shapes, icons, representation
    - Annotate activity nets
  - *ActivityNetEditor* for logical materials ActivityObj
    - An ActivityObj may have several visual representations (GraphObj)

```
                              ┌──────────┐
                         ◇────│ Lexicon  │
┌────────┐               │    └──────────┘
│        │◇──────────────┤    ┌──────────┐
│  Sane  │◇──────────────┼────│  Canvas  │
│        │◇──────────────┤    └──────────┘
└────────┘               │    ┌──────────────────┐
                         ◇────│ ActivityNetEditor │
                              └──────────────────┘
```
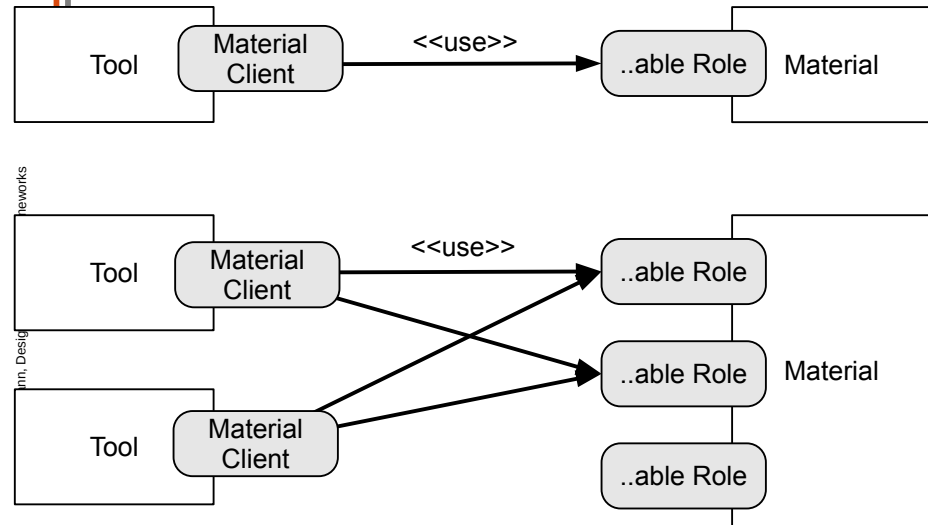
13.2 Tool Construction

Tool-Material Collaboration Pattern

- ▶ A *tool-material collaboration (*T&M role model, T&M access aspect) expresses the relation of a tool and the material
  - ▪ Characterizes a tool in the context of the material
  - ▪ The material in the context of a tool
  - ▪ The tool's access of the material. The tool has a view on the material, several tools have different views
- ▶ More specifically:
  - ▪ A *role* of the material, in collaboration with a tool
    - ▪ An interface of the material, visible by a tool, for a specific task
    - ▪ An abstract class
  - ▪ Roles of a material define the necessary operations on a material for one specific task
    - ▪ They reflect usability: how can a material be used?
    - ▪ Express a tool's individual needs on a material

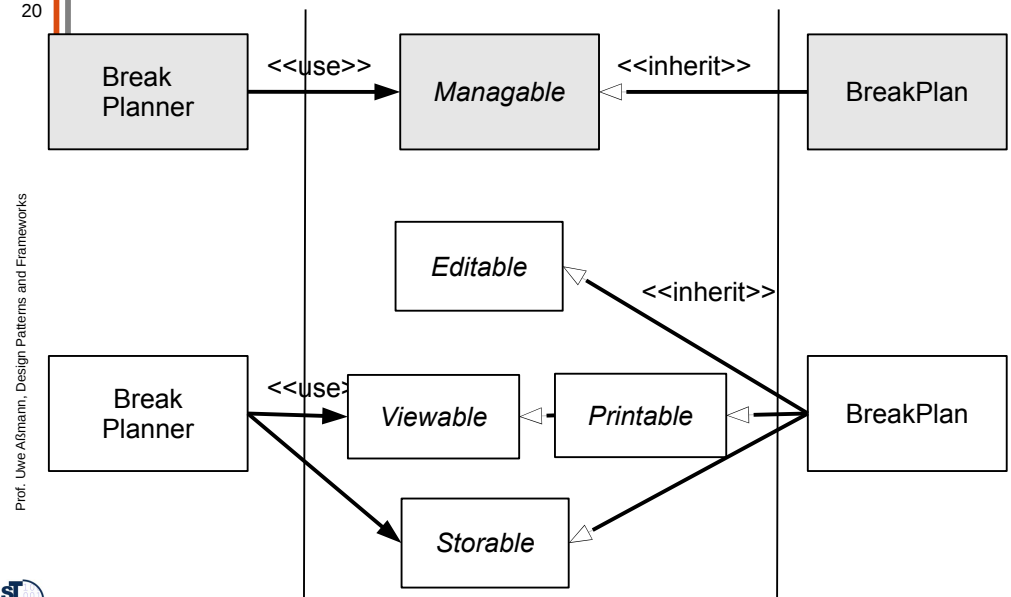*Prof. Uwe Aßmann, Design Patterns and Frameworks*

Tools and Their Views on Material

Implementing Tool-Material Roles With ..able-Interfaces

Prof. Uwe Aßmann, Design Patterns and Frameworks
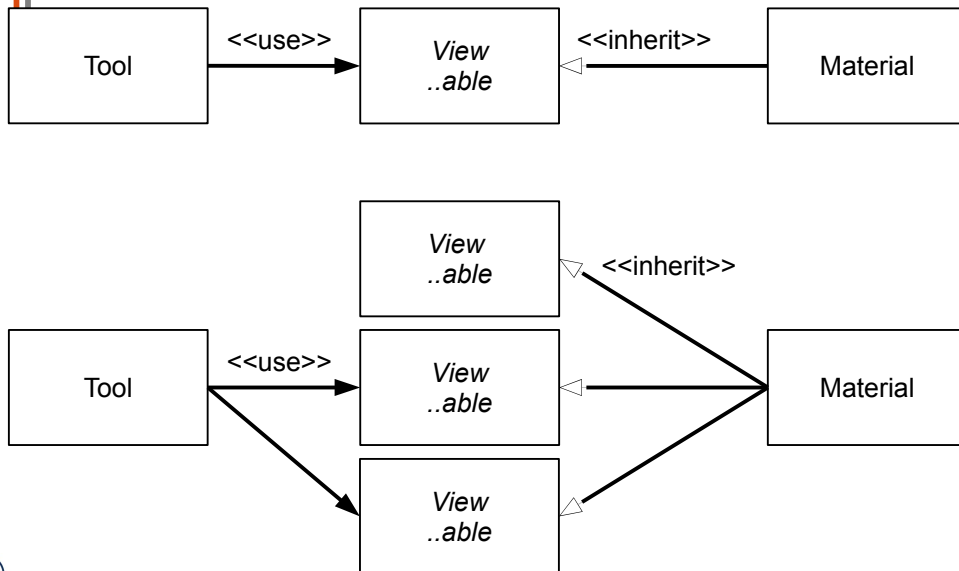
Tools/Views/Material with ..able-Interfaces

Prof. Uwe Aßmann, Design Patterns and Frameworks

Names of Roles

- ▶ The notion of a material-role helps a lot to understand the functionality of the materials
  - And helps to separate of them
- ▶ Often a "adjectified verb", such as Listable, Editable, Browsable, expresses the ability of a material from the perspective of a tool
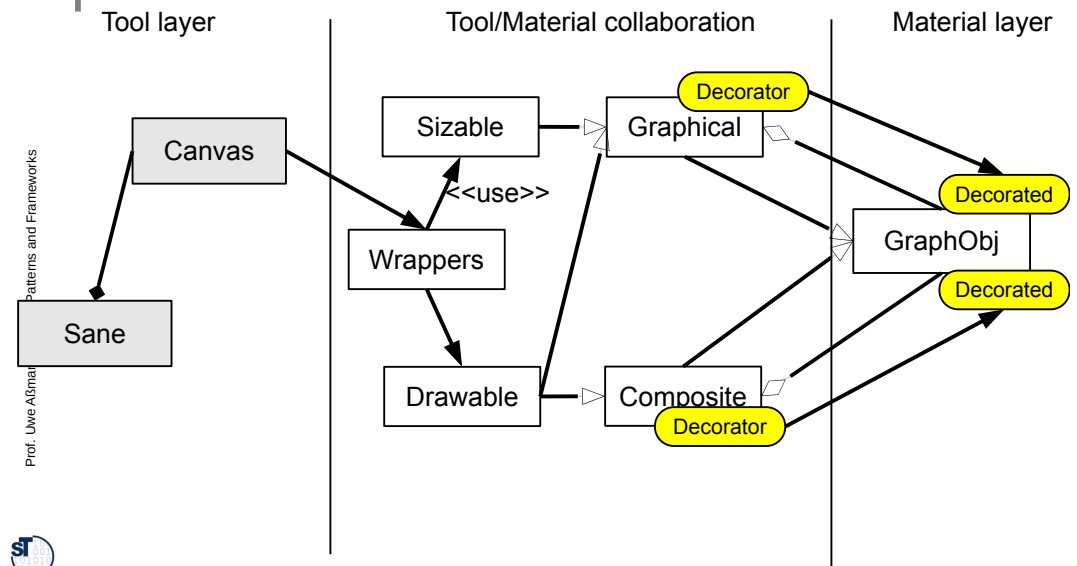
Prof. Uwe Aßmann, Design Patterns and Frameworks

Ex.: Access To Materials In TORA

- ▶ Access from tools to material via material-roles
  - Main tool: Storable
  - Canvas:
    - Drawable, Sizable with the help of wrappers DragWrapper, ResizeWrapper
    - Graphical role of GraphObj

Prof. Uwe Aßmann, Design Patterns and Frameworks

Alternative Implementations of Tool-Material Collaboration

- ▶ See chapter on role implementation
  - Construction of roles by interfaces
  - By multiple or mixin inheritance
- ▶ By ObjectAdapter pattern
- ▶ By Decorator pattern
- ▶ By Role-Object Pattern
- ▶ By GenVoca Pattern

Prof. Uwe Aßmann, Design Patterns and Frameworks
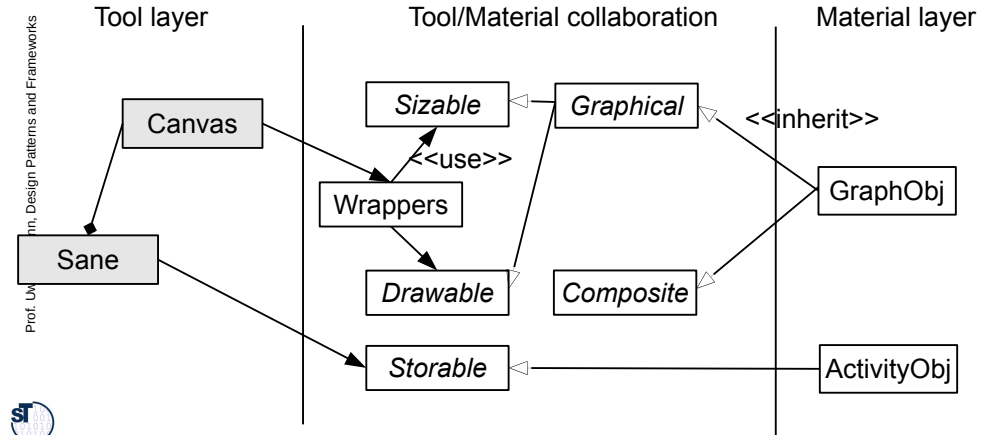
Ex.: Tools Accessing Material Via Decorators

- ▶ Converting roles into decorator objects

Prof. Uwe Aßmann, Design Patterns and Frameworks

Composition of a Tool and a Material Framework With Collaboration Roles

▶ Since Material-roles are roles, Tool layer and Material layer can be modeled as frameworks (which then can be composed by role composition/use)
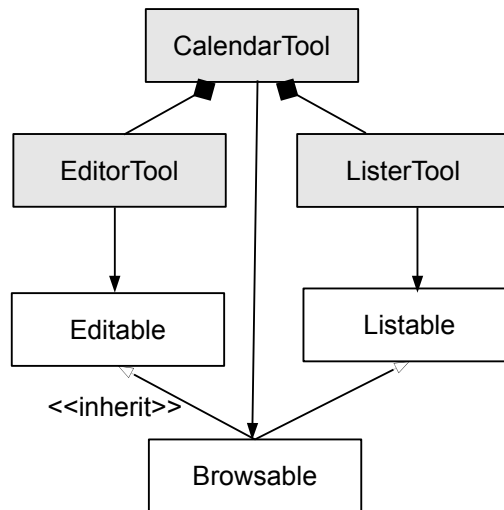

Tool Framework — Subtool → Wrapper (Graphics Client, Drawing Client), Main (Storable Client)

Material Framework — Graphical, Sizable, Drawable, Composite (GraphObj); Storable (EditObj)

(vertical text) Prof. Uwe Aßmann, Design Patterns and Frameworks

---
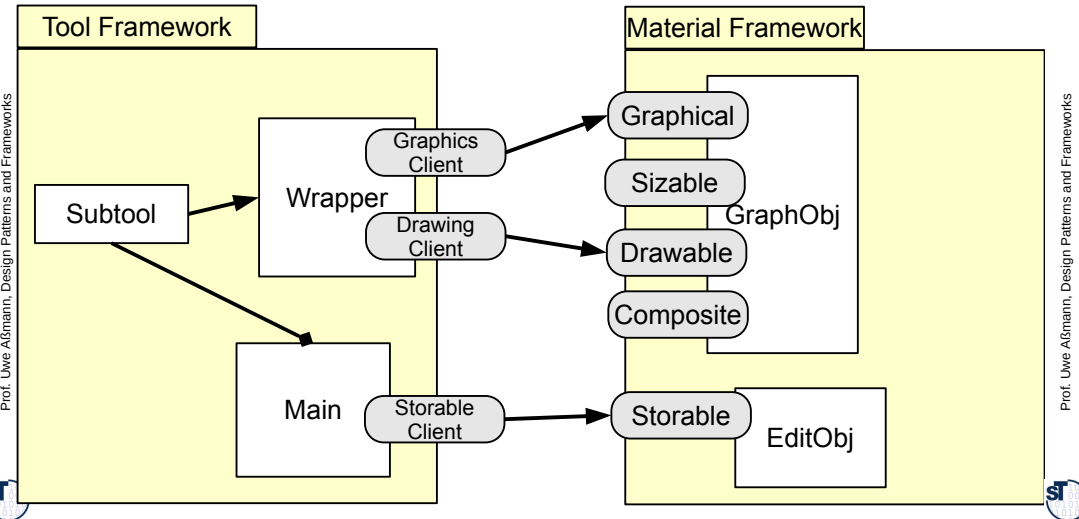
Tool Construction: Structured Tool Pattern

▶ Structured tools
  ▪ Atomic tools
  ▪ Composed tools (with subtools)
  ▪ Recursively composed tools (Composite pattern)
▶ Structured along the tasks
▶ A complex tool creates, delegates to, and coordinates its subtools

(vertical text) Prof. Uwe Aßmann, Design Patterns and Frameworks

---

Tool Construction: Structured Tool Pattern

▶ Subtools are aggregated
▶ A subtool can work on its own material
  ▪ Or on the same material as a supertool, but with fewer or less complex roles
▶ Advantage: complex tools see complex roles, simple tools simple roles
▶ The role hierarchy opens features of the material only as needed (good information hiding)

CalendarTool → EditorTool, ListerTool; EditorTool → Editable; ListerTool → Listable; Editable, Listable → Browsable

<<inherit>>

(vertical text) Prof. Uwe Aßmann, Design Patterns and Frameworks

---

Tool Construction: Composite as Structured Tool Pattern

▶ The Composite pattern can be used to build up recursive tools

Tool, CompositeTool, AtomicTool, TableTool, TableCellTool, AtomicCellTool, Zoomable, Listable, Browsable

(vertical text) Prof. Uwe Aßmann, Design Patterns and Frameworks
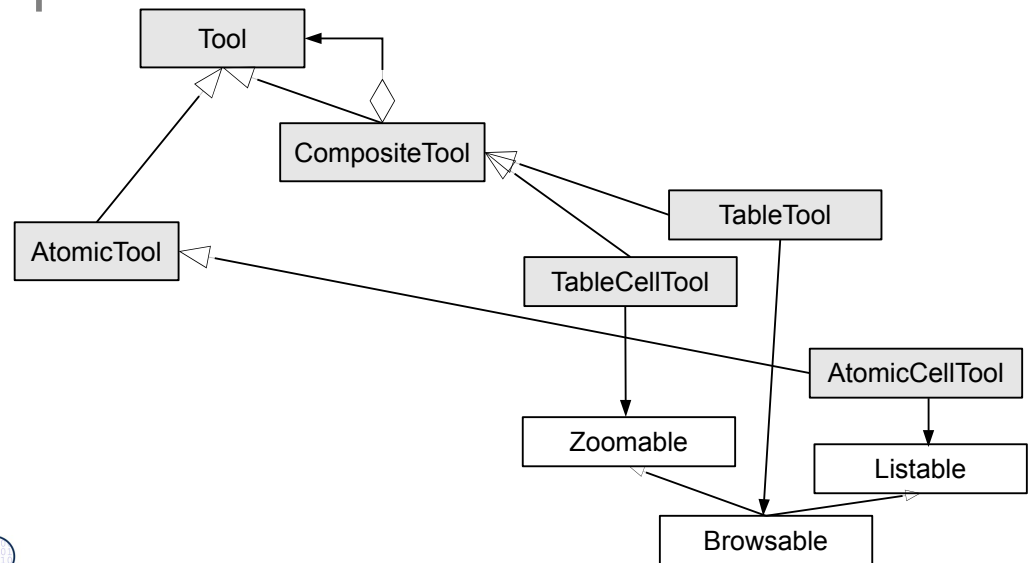
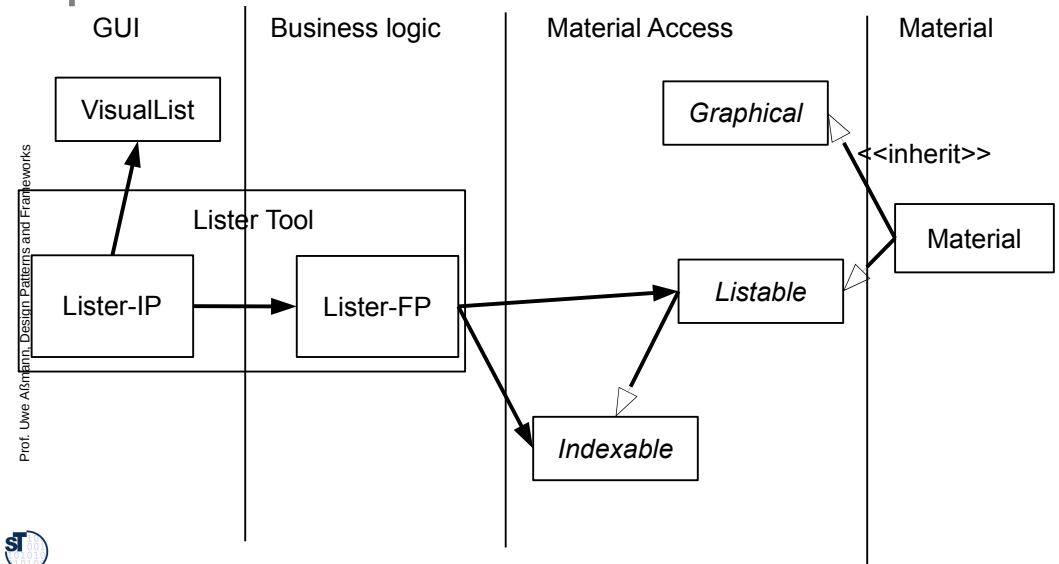# Tool Construction: Separation of Function and Interaction

- ▶ Separation of function and interaction
  - Separation of user interface and application logic, as in 3-tier
  - Tools have one functional part and one or several interaction part
- ▶ Functional Part (FP):
  - Manipulation of the material
  - Access to Material via material-roles
- ▶ Interaction Part (IP):
  - Reactive on user inputs
  - Modeless, if possible
 - Can be replaced without affecting the functional part

Prof. Uwe Aßmann, Design Patterns and Frameworks

Interaction Part (IP) and Functional Part (FP)

- ▶ FP create a new layer

Prof. Uwe Aßmann, Design Patterns and Frameworks

How TORA Tools Access Their Material

- ▶ Tool Sane is split into IP and FP
  - Manages a frame on the screen for drawing

Prof. Uwe Aßmann, Design Patterns and Frameworks
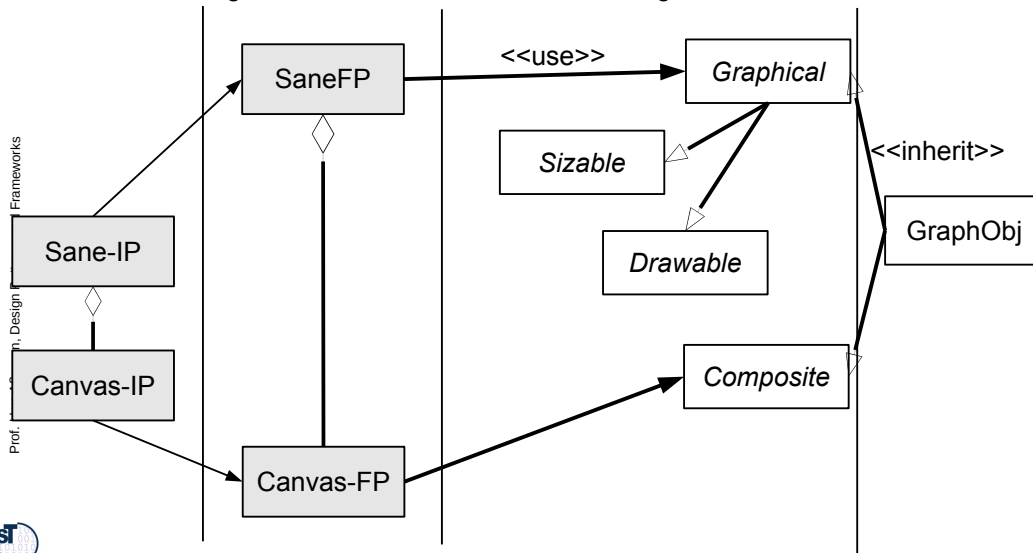
IP-FP TAM Refines MVC

- ▶ Tools contain
  - a view (IP)
  - the controller (FP)
  - and the managing part of the model
- ▶ The model is split between tool-FP, material access, and material

Prof. Uwe Aßmann, Design Patterns and Frameworks

Coupling between Function and Interaction With Observer

- ▶ Play-Out via Observer pattern: IP listen to FP changes and actions
- ▶ Play-In via call
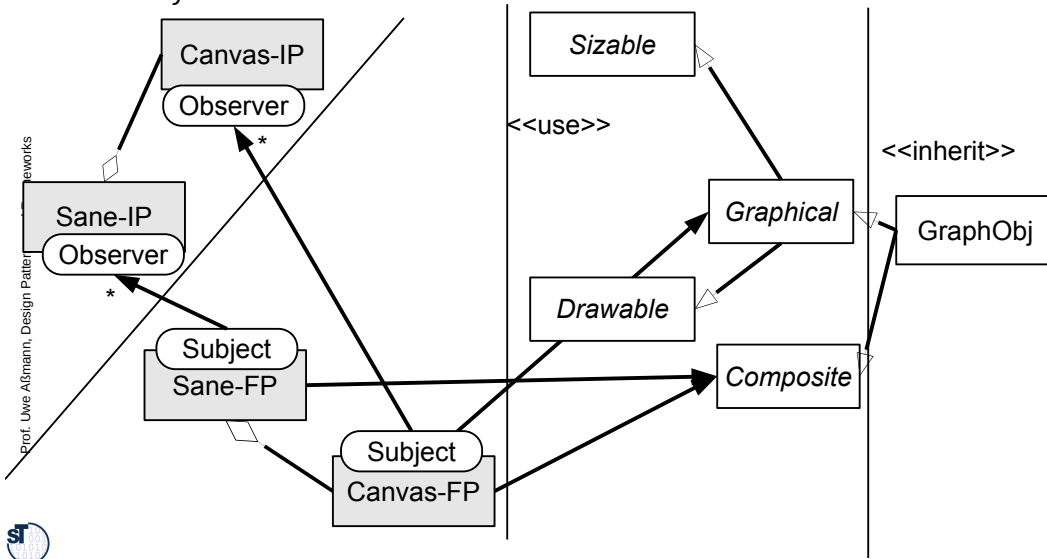
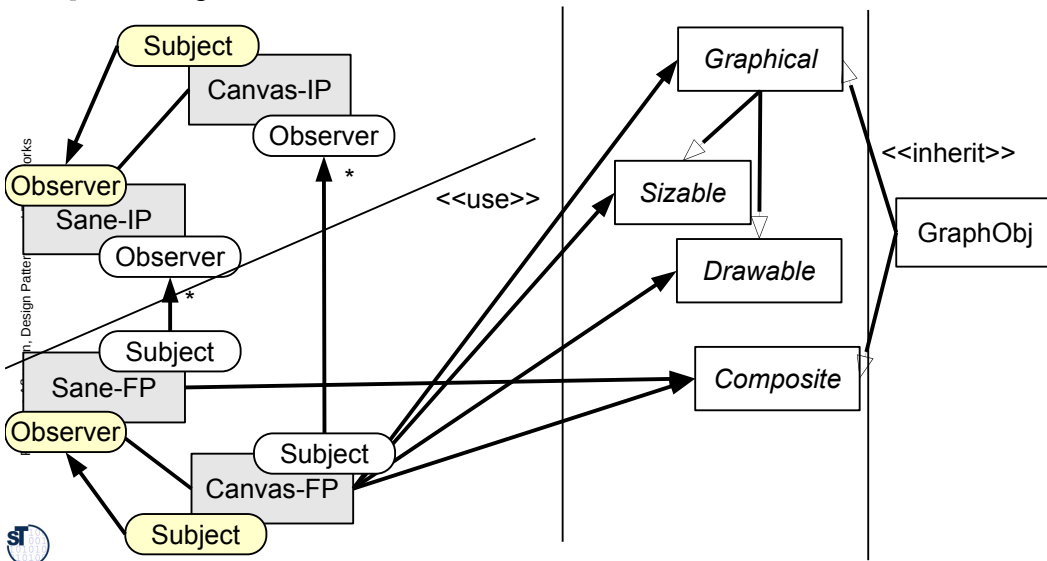# Coupling between Subtool-FP and Supertool-FP

- ▶ **Vertical tool decomposition** by structuring into subtools with Bridge, Composite, Bureaucracy
- ▶ **Horizontal tool decomposition** into IP and FP
- ▶ How to add new subtools at runtime?
  - ▪ Decomposition should be extensible
    - ▪ Vertically: for Composite, this is the case
    - ▪ Horizontally, Observer serves for extensibility
  - ▪ Communication should be extensible (next slide)

Symmetric Coupling between Subtools and Supertools by Observer

- ▶ Vertical Observer: Supertools are notified from subtools if something changes

Coupling between Subtools and Supertools By Symmetric Bureaucracy

- ▶ IP and FP hierarchy can work with a Bureaucracy each
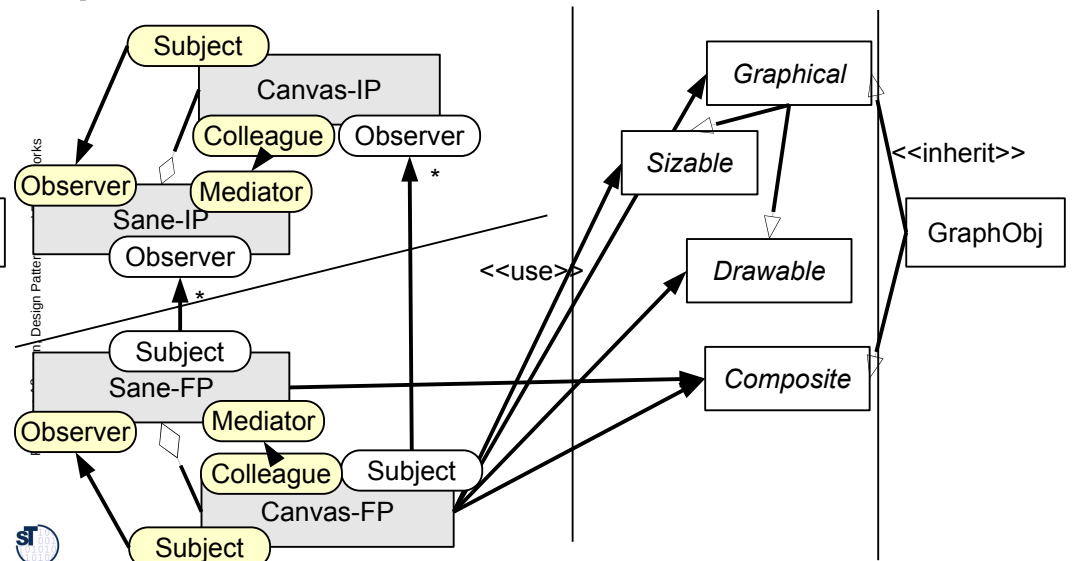
Creation of New Subtools

- ► Initiated by a Super-FP, which decides to create a new sub-FP
- ► Steps:
  - ▪ Super-FP notifies Super-IP
  - ▪ Super-IP may create one or several sub-IP
    - ▪ Connects them as observers to the sub-FP
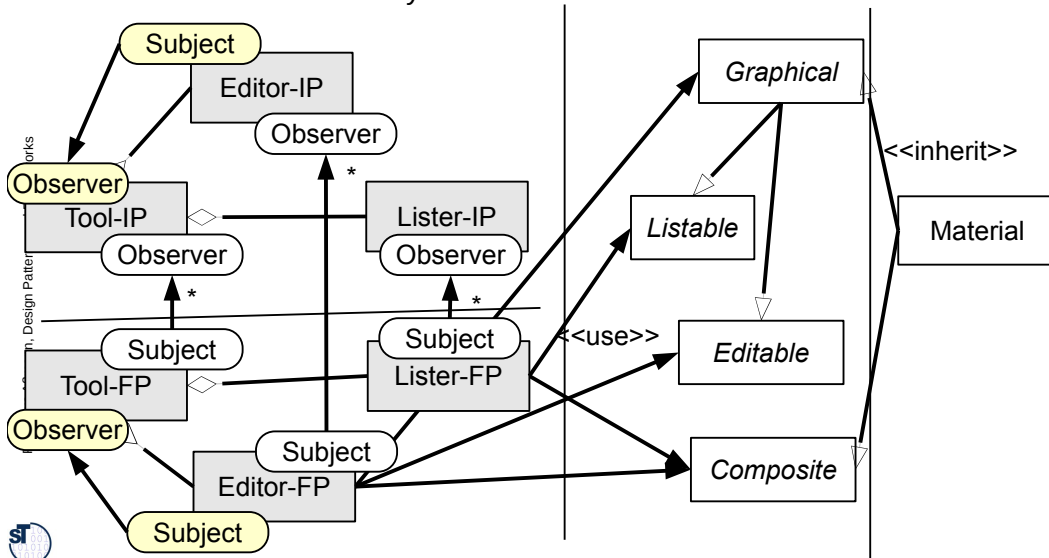
Prof. Uwe Aßmann, Design Patterns and Frameworks

Non-Symmetric Coupling between Subtools and Supertools

- ► Super-IPs can be notified by Super-FPs
- ► Optimization: Several of the event channels can be coalesced for better runtime behavior
  - ▪ Merging FP and IP again, getting rid of Observer, but no extensibility anymore
  - ▪ Substituting events by hard-coded calls

Prof. Uwe Aßmann, Design Patterns and Frameworks
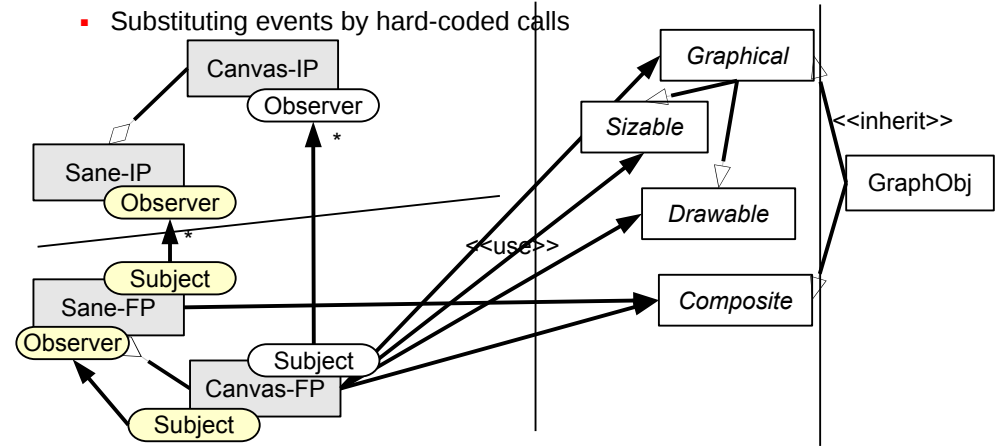
Example:
Generic Editor and Lister Framework

- ► Supertools are notified from subtools if something changes
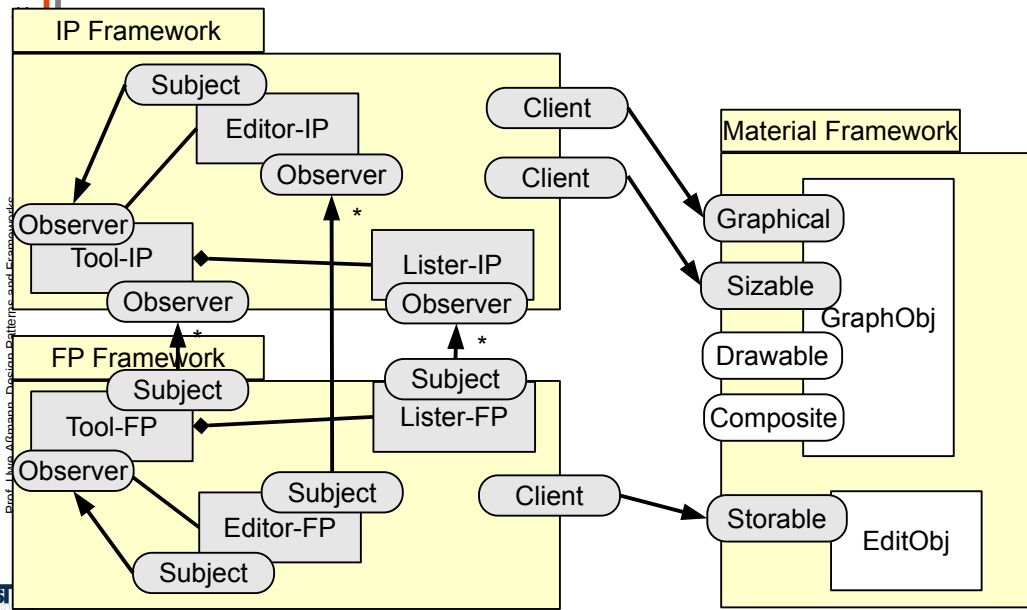- ► Can be used for every editor and lister of material

Instantiated to a Calendar Editor and Lister Tool

- ► Supertools are notified from subtools if something changes
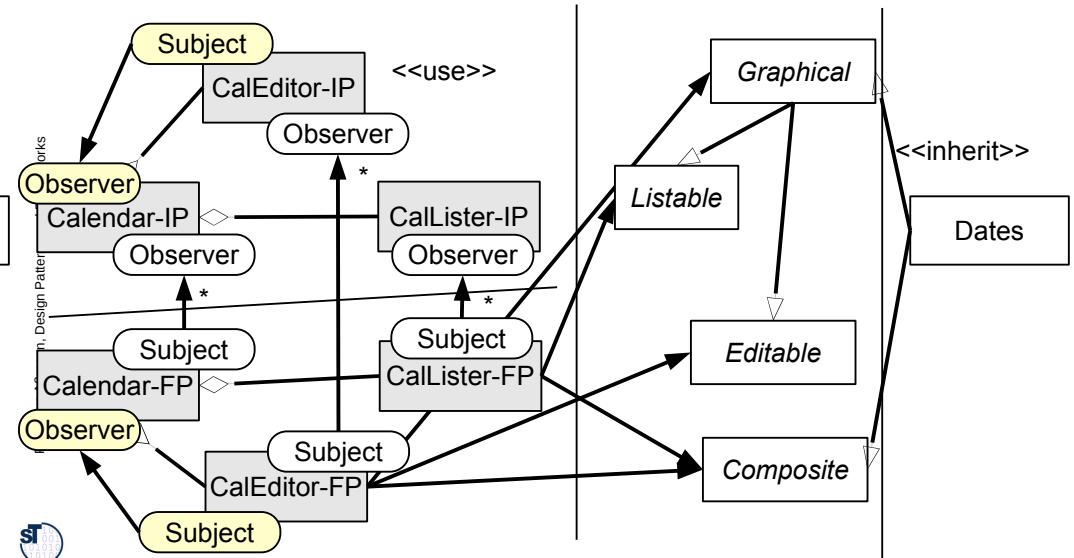
The Generic Editor in Framework Notation

IP Framework

Subject
Editor-IP
Observer

Observer
Tool-IP
Observer

Lister-IP
Observer

Client
Client

Material Framework

Graphical
Sizable
GraphObj
Drawable
Composite

FP Framework

Subject
Tool-FP
Observer

Subject
Editor-FP
Subject

Subject
Lister-FP

Client

Storable
EditObj

13.3 Environment

The Environment

▶ Tools and Materials live in an environment with
- Tool coordinators
- Material administrations
- Event coordinators

▶ The environment initializes everything, displays everything on the desktop, and waits for tool launch

Environment

ToolCoordinator   MaterialAdministration   EventCoordinator

Tools   Material   Events

* * *
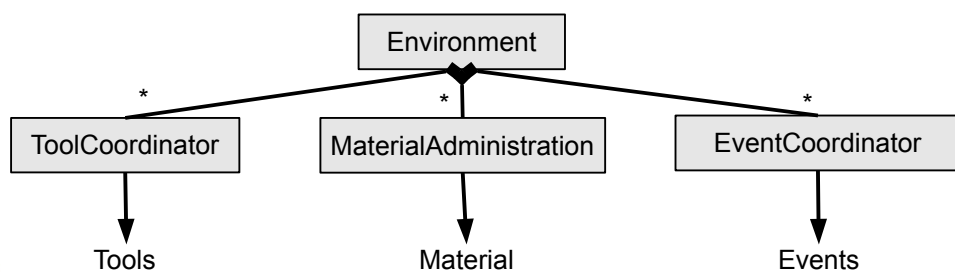
Tool Coordinator in the Environment

▶ The **tool coordinator** is a global object
- Groups a set of tools and their related material
  - Contains
    - A Tool-Material dictionary of all tools and the materials they work on
    - A tool factory

▶ Is a Mediator between FPs and other tools
- Usually, FPs talk to their supertools and their related IPs. When materials depend on other materials in complex ways, other tools have to be informed
- The ToolCoordinator uses the Tool-Material dictionary to notify tools appropriately

Example: TORA Tool Coordinator

# 13.3.1. Pattern: Constrained Material Container

Design Patterns and Frameworks, © Prof. Uwe Aßmann
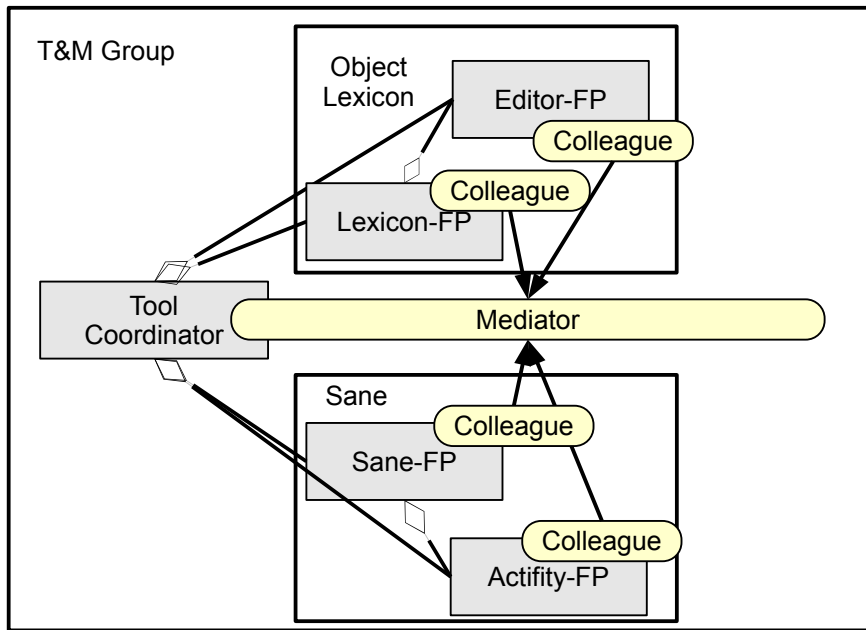
Problem: Dependencies Among Materials

► Materials may depend on each other
► Example MeetingScheduler
  ▪ Maintains regular meeting dates (week, month, year)
  ▪ Should collaborate with the Calendar tool that maintains individual dates
► Clearly, these materials are dependent on each other
  ▪ The Calendar tool should take in meetings as individual dates
  ▪ The MeetingScheduler should block meetings if individual dates appear in the calendar

Pattern: Constrained Material Container
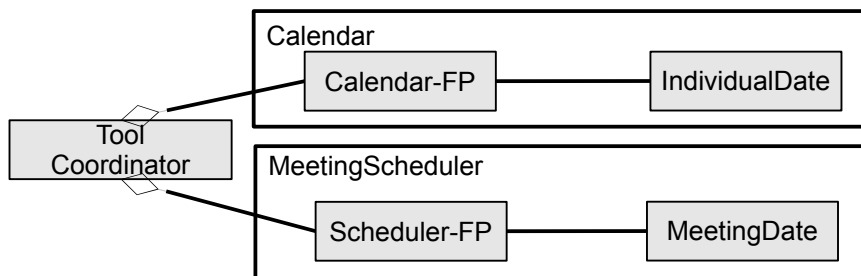
► We group all material that depend on each other into one *Material container*
  ▪ And associate a *constraint object* InSaneConstraint that maintains the dependencies
 ▪ The constraint object is a Strategy for controlling the dependencies of the Material
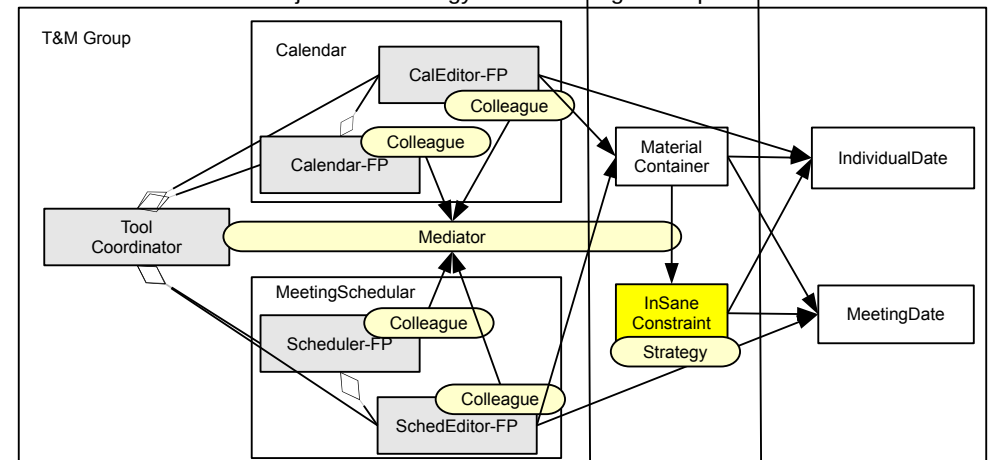
Tool Coordinator and Material Container

▶ Unfortunately, Constrained Material Containers of the group must query the dictionary of the Tool Coordinator,
  ▪ to know about the currently available tools, to activate constraints
  ▪ (which introduces an ugly dependency between them...)

Example: How TORA Tools Access Their Material
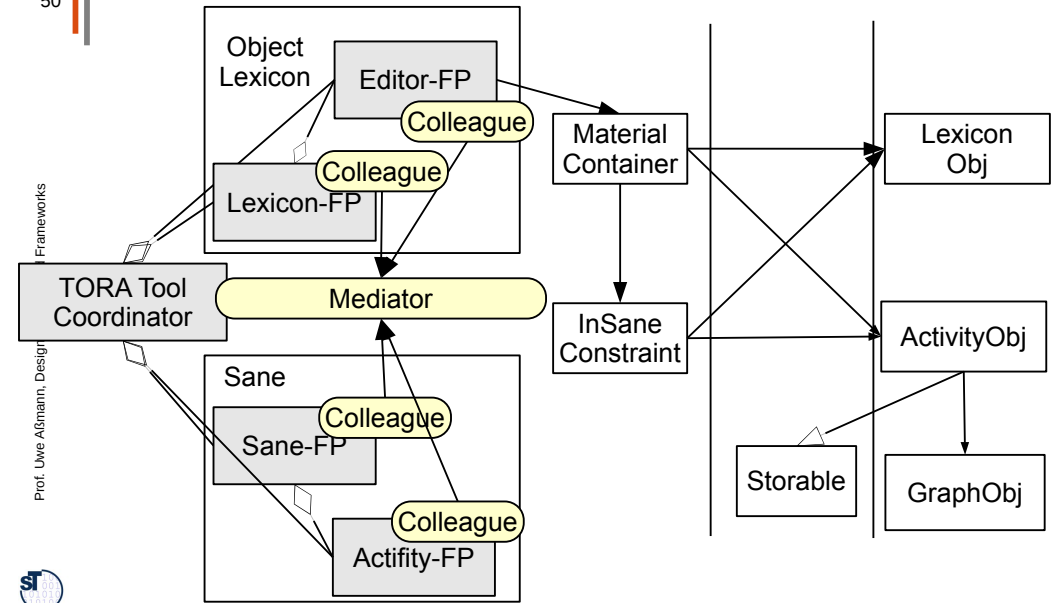
TORA Material Constraints

▶ For each ActivityObj, there is a LexiconObj
  ▪ The user can textually edit the LexiconObj to document the ActivityObj and the GraphObj
▶ All Materials are in a MaterialContainer
  ▪ Uses a ConstraintObject InSaneConstraint to make sur that the label of the ActivityObj is always the same as that of the LexiconObj
▶ If an ActivityObj is created, deleted, or changed, the tool coordinator is informed
  ▪ And informs all related tools of TORA
 ▪ The tool coordinator is a mediator

Automaton

▶ An *automaton* (*interpreter, workflow engine*) is an automated tool for repeated tasks
  ▪ Similar to a macro-tool, a variant of Design Pattern Interpreter
  ▪ Can run in the background
  ▪ Often realized as separate machine processes
▶ An automaton encapsulates an automated *workflow* (or *process*)
  ▪ Production of a complex artifact
  ▪ Storing a complex technical object
  ▪ Producing data in different versions
▶ Workflow can be specified by statecharts, activity diagrams, data-flow diagrams, Petrinets, workflow languages

An Automaton Booking Calendar Dates

▶ The Automaton books regular meetings as dates into the calendar
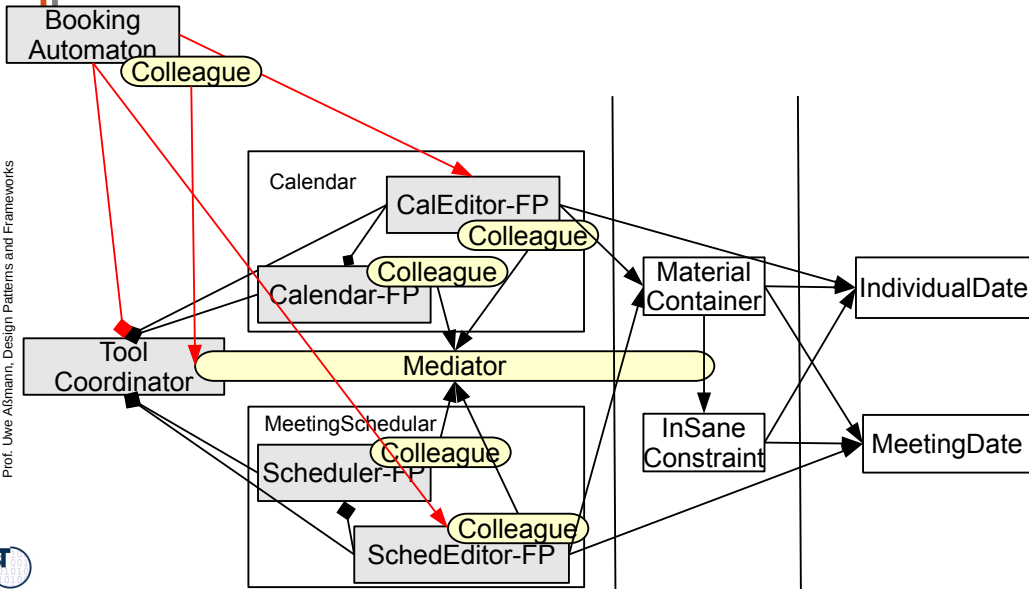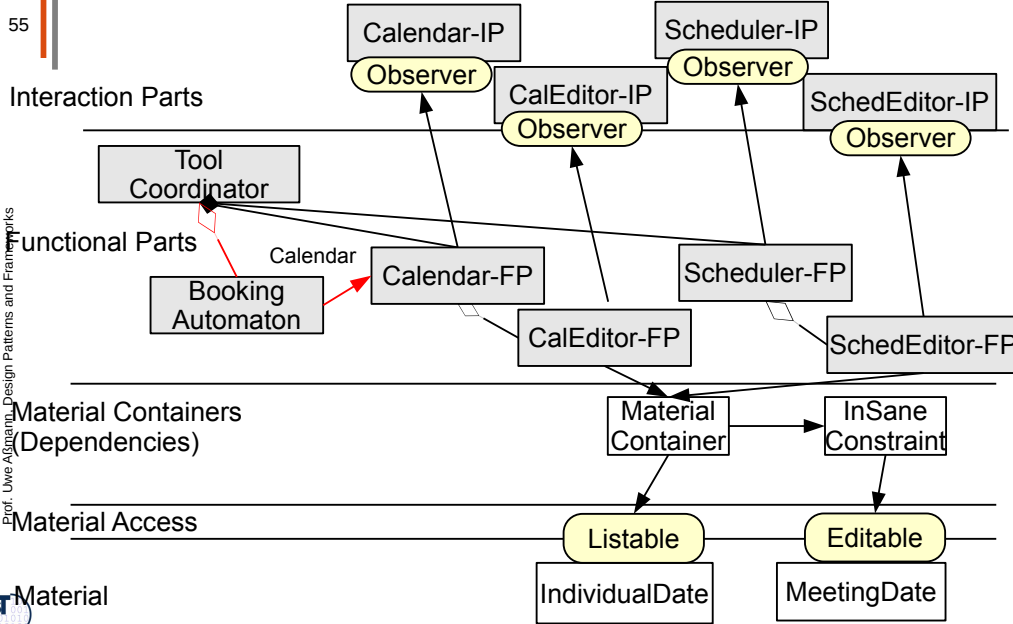

13.4 TAM and Layered Frameworks

Now, let's order the patterns of TAM into layers

What happens?

Design Patterns and Frameworks, © Prof. Uwe Aßmann
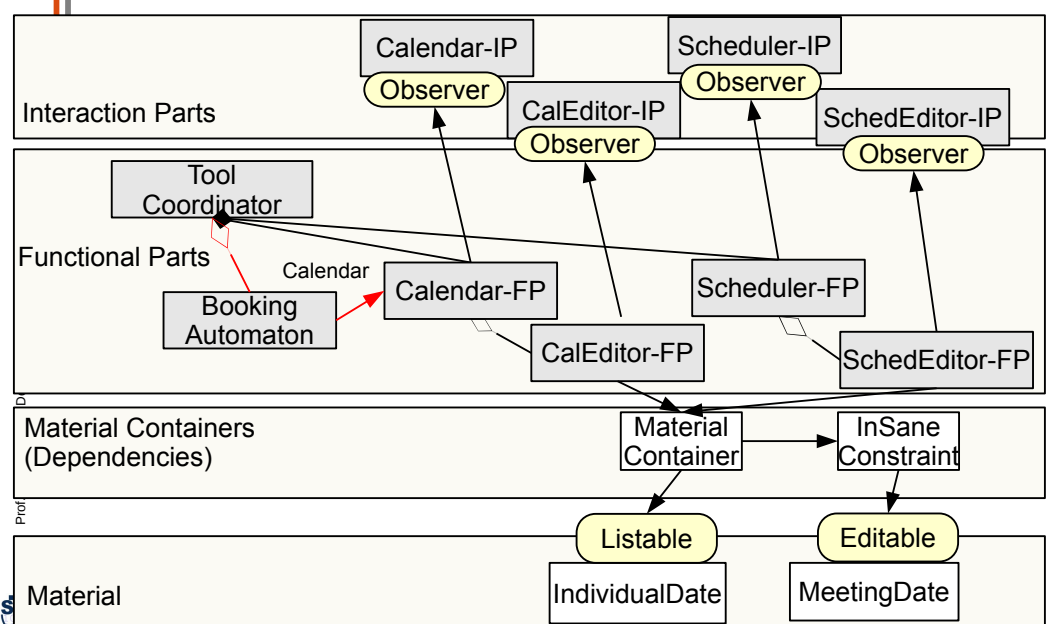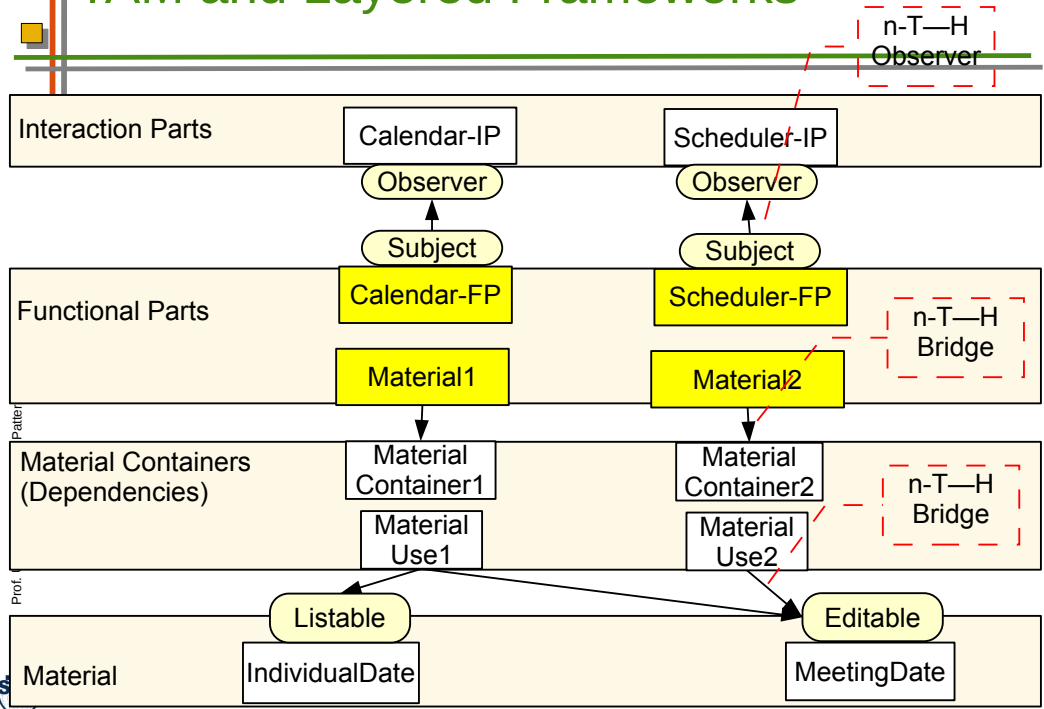

TAM and Layered Frameworks

TAM and Layered Frameworks

TAM and Layered Frameworks

TAM Is a Variant of a Layered Framework

► Combining different miniconnectors between the layers
  ▪ n-T—H Observer between IP and FP
  ▪ n-T—H Bridge between FP and MaterialUse
  ▪ n-T—H Bridge between MaterialUse and Material, with roles as access for material
► Hence, interactive applications can be seen as instances of a layered framework
  ▪ That uses not only RoleObject as mini-connectors, but also Observer and Bridge.
  ▪ Hence the analogy to 3-tier
► This gives hope that we can construct layered frameworks for interactive applications in the future!

Summary

► The T&M conceptual pattern is a very important pattern for object-oriented development; all classes fall into these categories
► T&M is a pattern language for constructing interactive applications
  ▪ Refines 3-tier and MVC
  ▪ Uses Command, Strategy, Observer, Composite, etc.
  ▪ Defines several new complex patterns such as Separation of IP and FP
► TAM is a variant of a layered framework, using n-T—H miniconnectors (Observer, Bridge) between the layers
  ▪ Pree's framework hook patterns play an important role

The End

Prof. Uwe Aßmann, Design Patterns and Frameworks