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

Prof. Dr. U. Aßmann
Software Technology Group
Department of Computer Science
Technische Universität Dresden
WS 17/18 - Jan 22, 2018

Lecturer: Dr. Sebastian Götz

Literature

- 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/

Secondary Literature

- Heinz Züllighoven et al. **The object-oriented construction handbook.** Morgan Kaufmann Publishers, 2004.
- Heinz Züllighoven et al. **Das objektorientierte Konstruktionshandbuch – nach dem Werkzeug und Material-Ansatz.** Dpunkt-Verlag, Heidelberg, 1998. (*german*)
- Dirk 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.
- 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

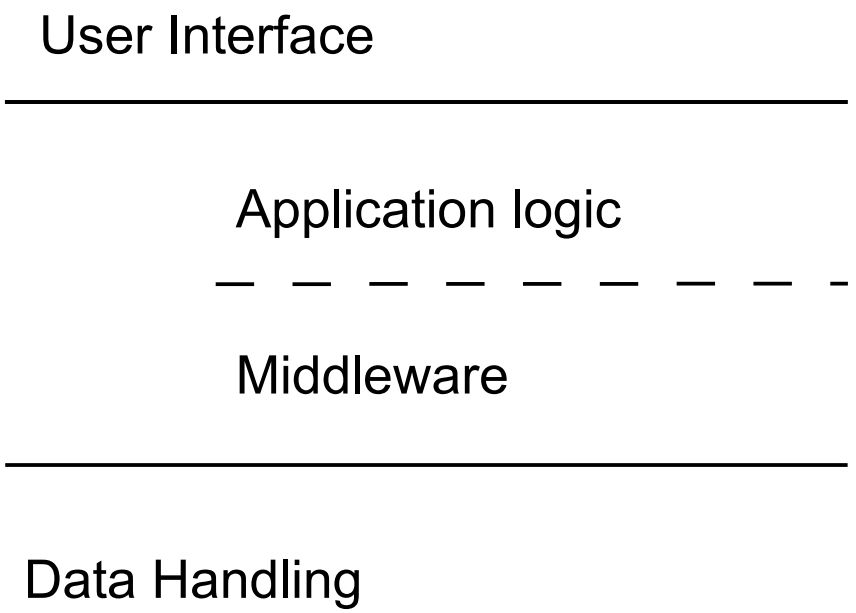
14.1 Elements of "Tools and Materials"

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 the 3-tier architecture
- However, the 3-tiers are about structuring the application logic
- The tools and materials metaphor fits as an abstraction for user interaction



Materials

- 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 with value semantics)

- 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

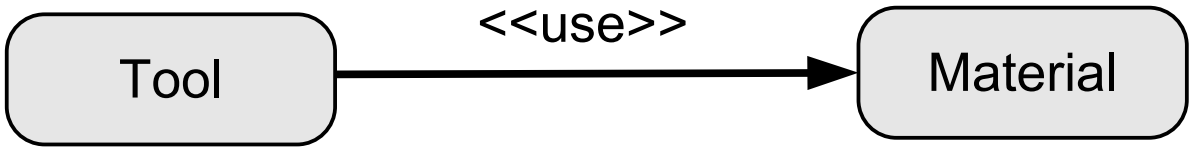
- Active entitites
  - Tools are means of work. They embody the experience of how to work with material
  - Tools have a view on the material (i.e., only “see” what is required for their purpose).
  - Often visible on the desktop as wizards, active forms,..
  - Tools give feedback to the user
  - Tools have a state
- If well-designed, they are transparent, light-weight, and orthogonal to each other
- Examples:
  - Browser – Contents of a folder, websites
  - Interpreter – Code and data
  - Calendar – Appointments
  - 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
- In implementations, tools are a often realized as a variant of the Command (i.e., 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
- We work with different tools on the same material

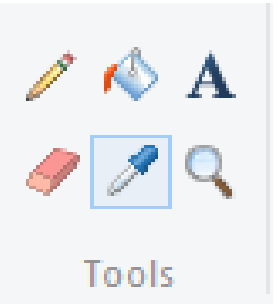


(Work-)Environment

- 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

Example: Microsoft Paint

- Tool to work with images
- Comprises several tools
  - Cropping
  - Drawing lines, circles, rectangles, …
  - Filling areas
  - Etc.
- Paint is the supertool

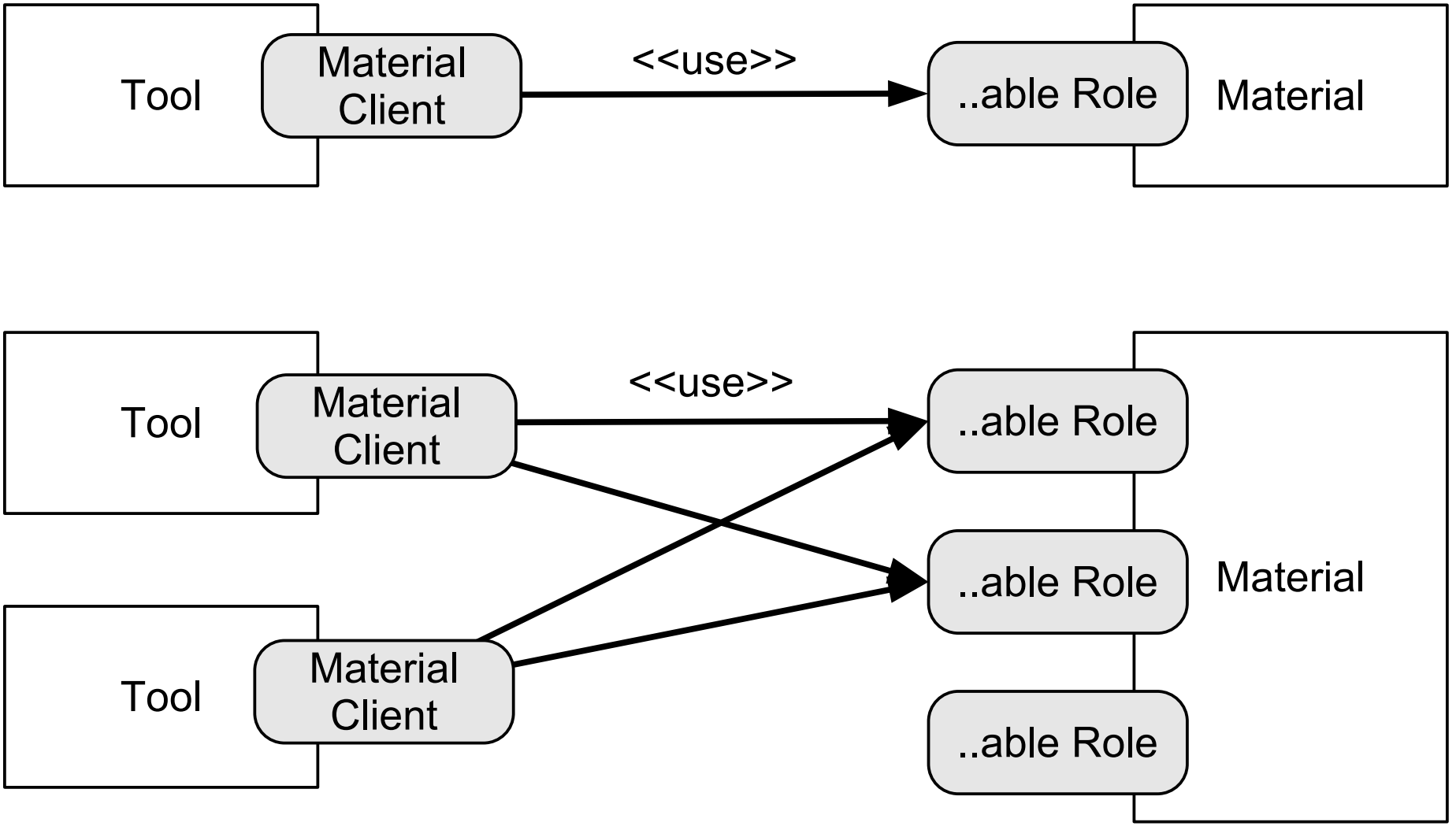


14.2 Tool Construction

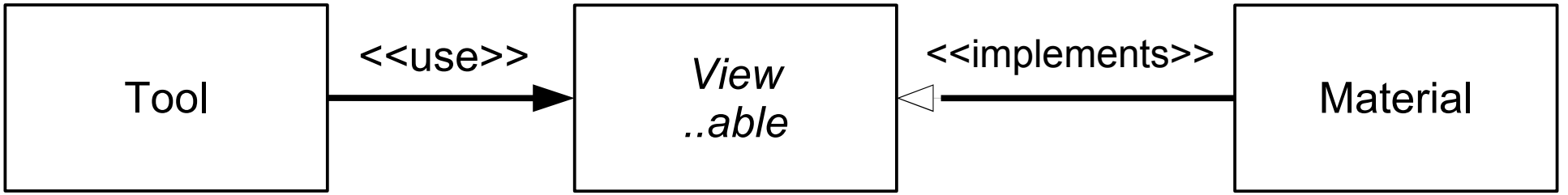
Tool-Material Collaboration Pattern

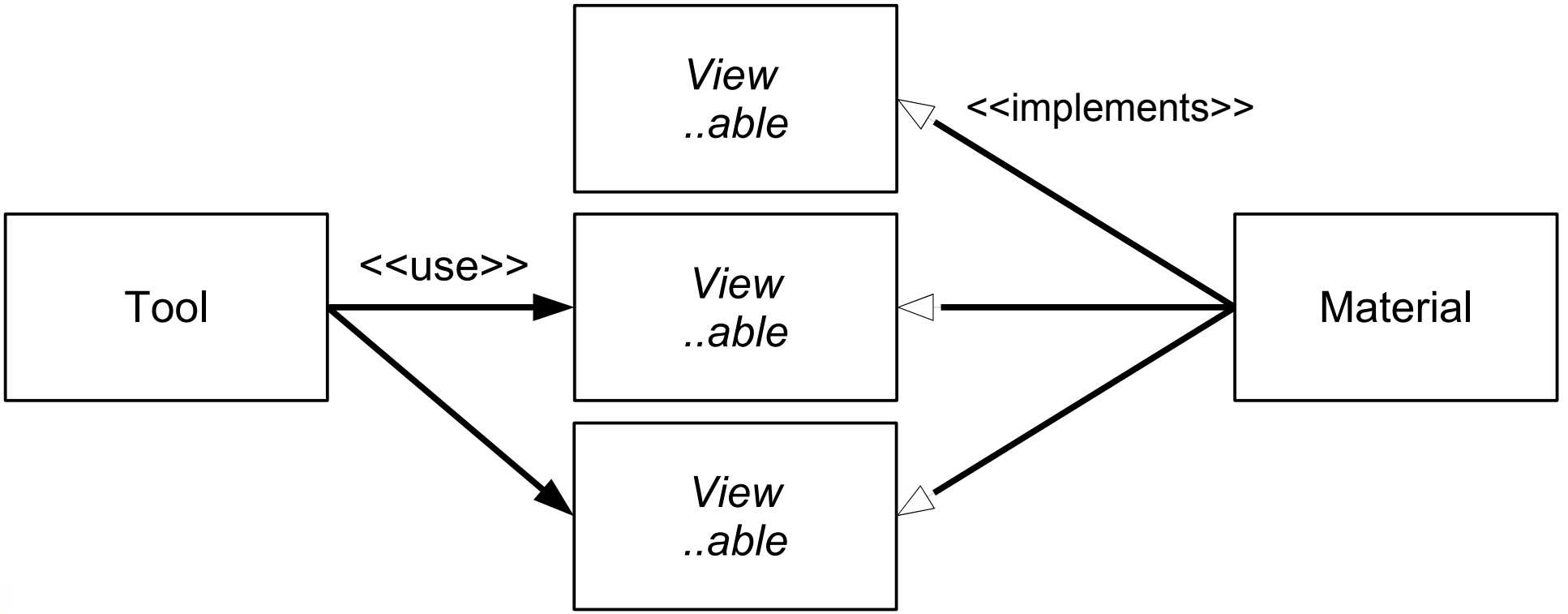
- A *tool-material collaboration (*T&M role model, T&M access aspect) expresses the relation of a tool and the material
  - Characterizes a tool in the context of the material
  - The material in the context of a tool
  - The tool's access of the material. The tool has a view on the material, several 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
  - Roles of a material define the necessary operations on a material for one specific task
    - They reflect how a material can be used
    - Express a tool's individual needs on a material

Tools and Their Views on Material

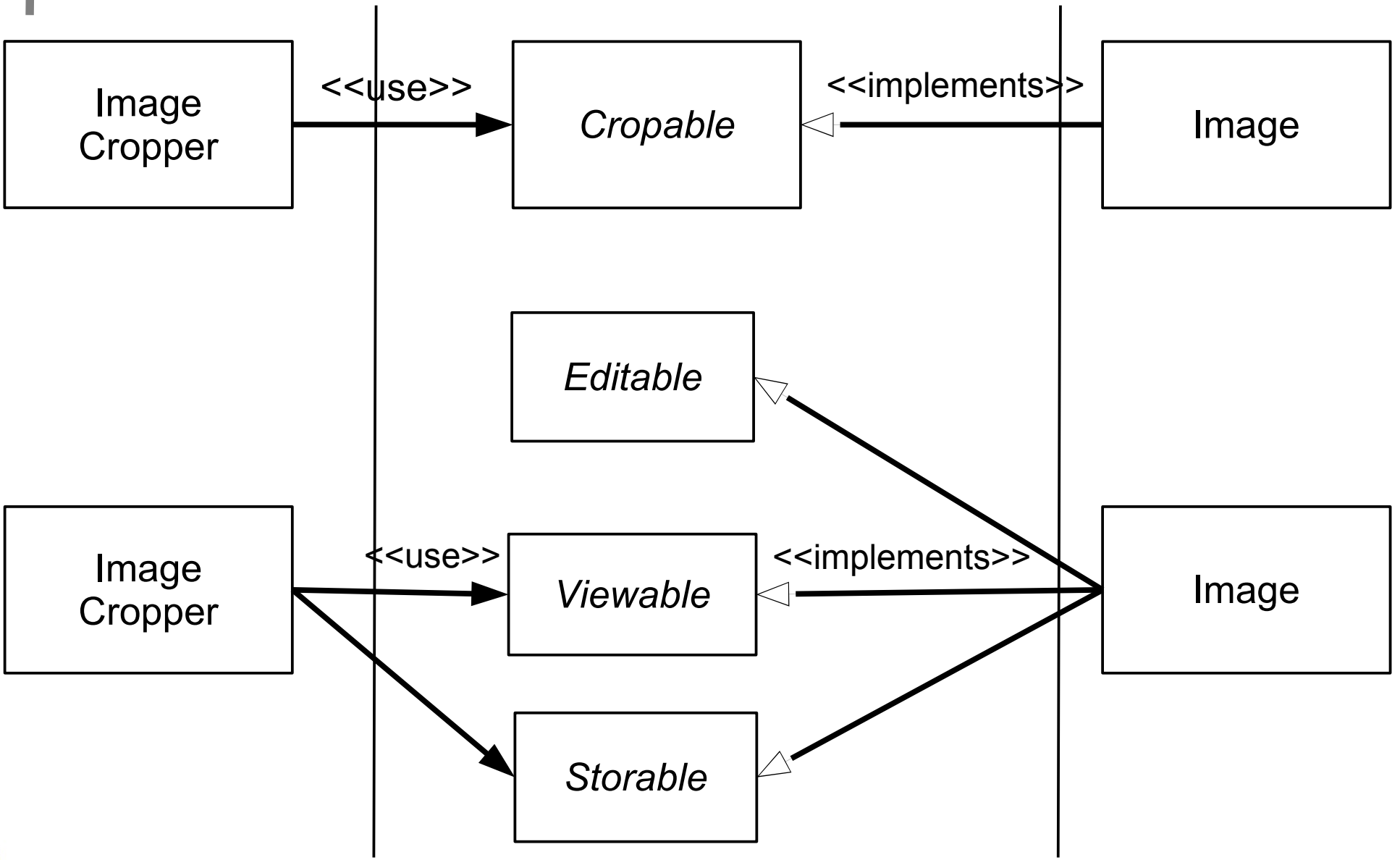


Tools/Views/Material with Interfaces





Tools/Views/Material with Interfaces

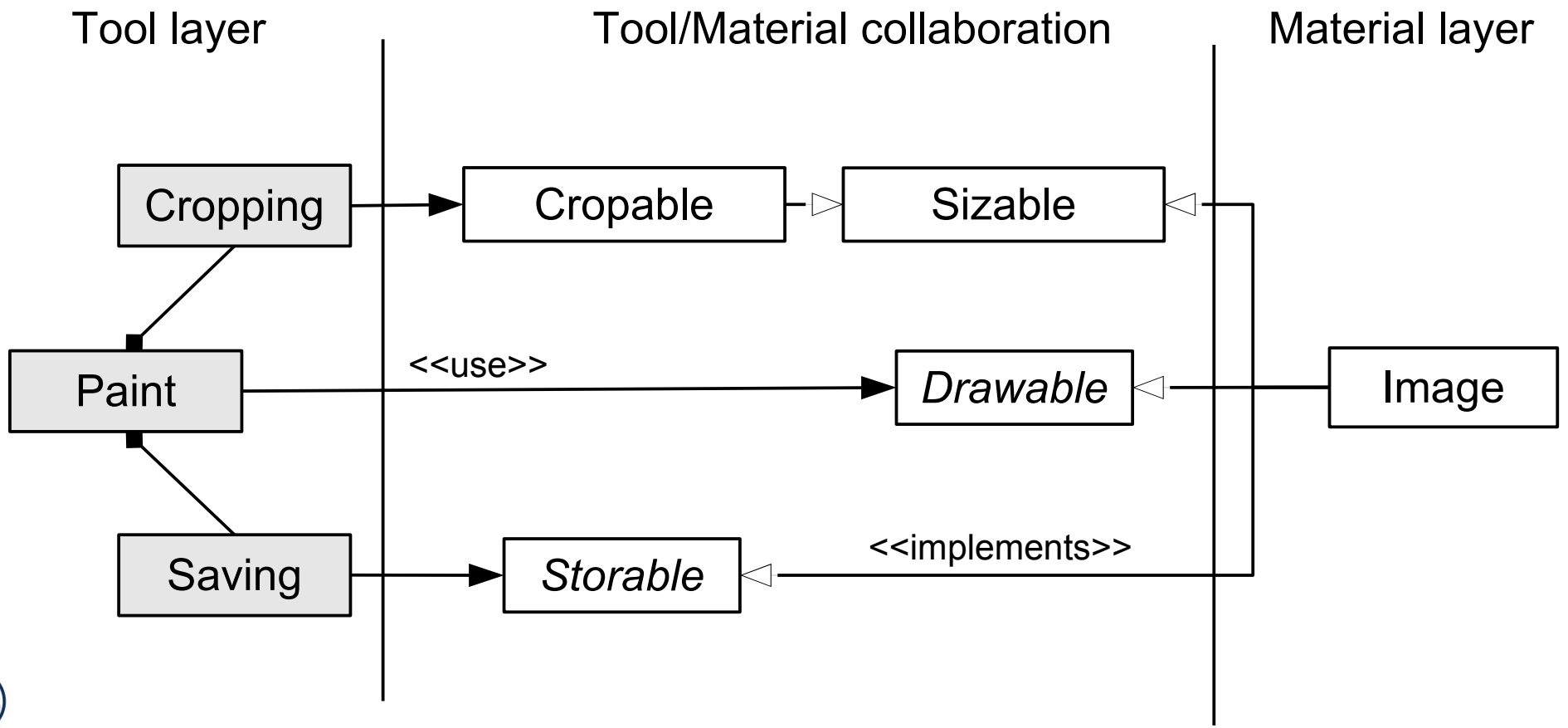


Names of Roles

- The notion of a material-role helps a lot to understand the functionality of the materials
  - And helps to separate them
- Often an “adjectivized verb”, such as Listable, Editable, Browsable, expresses the ability of a material from the perspective of a tool

Ex.: Access To Materials In Paint

- Access from tools to material via material-roles
  - Main tool Paint: Drawable
  - Tool Cropping: Cropable via Sizable
  - Tool Saving: Storable

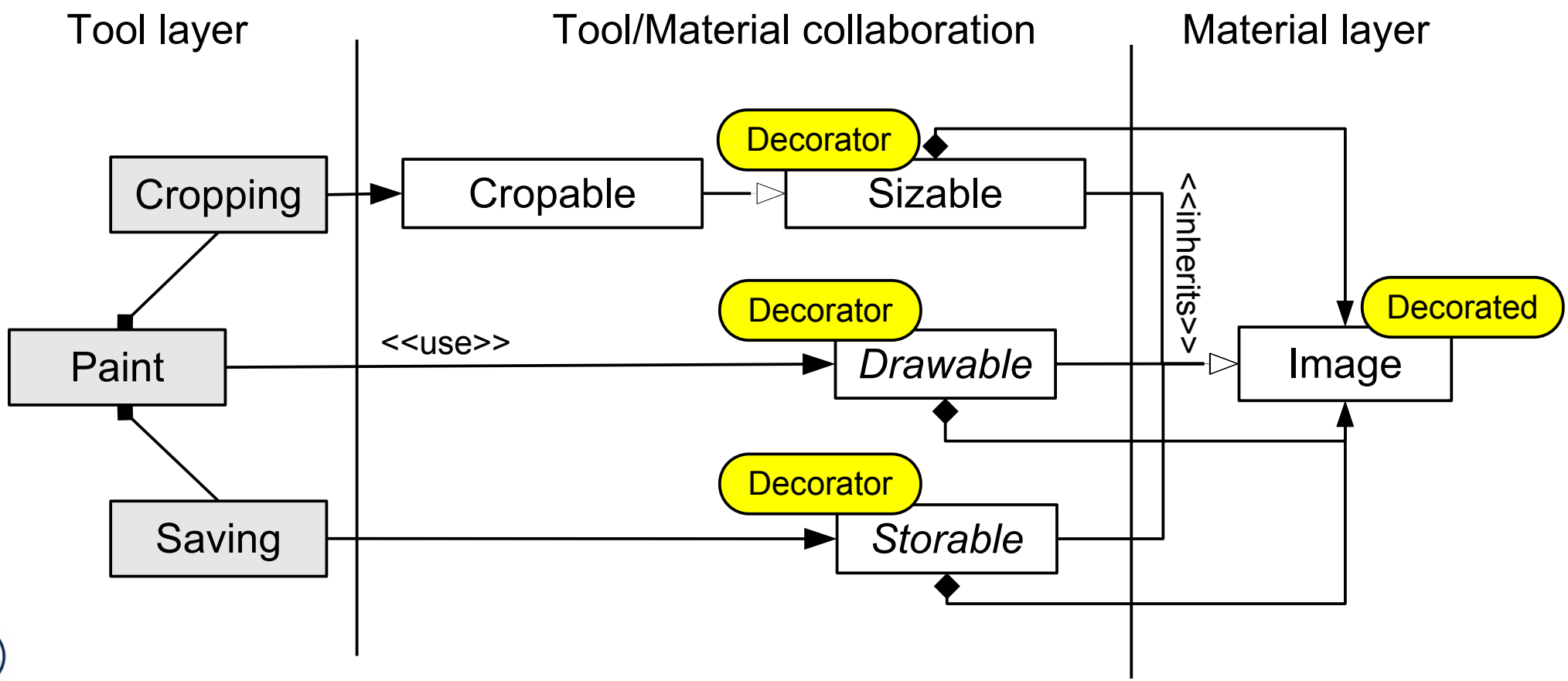


Alternative Implementations of Tool-Material Collaboration

- 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

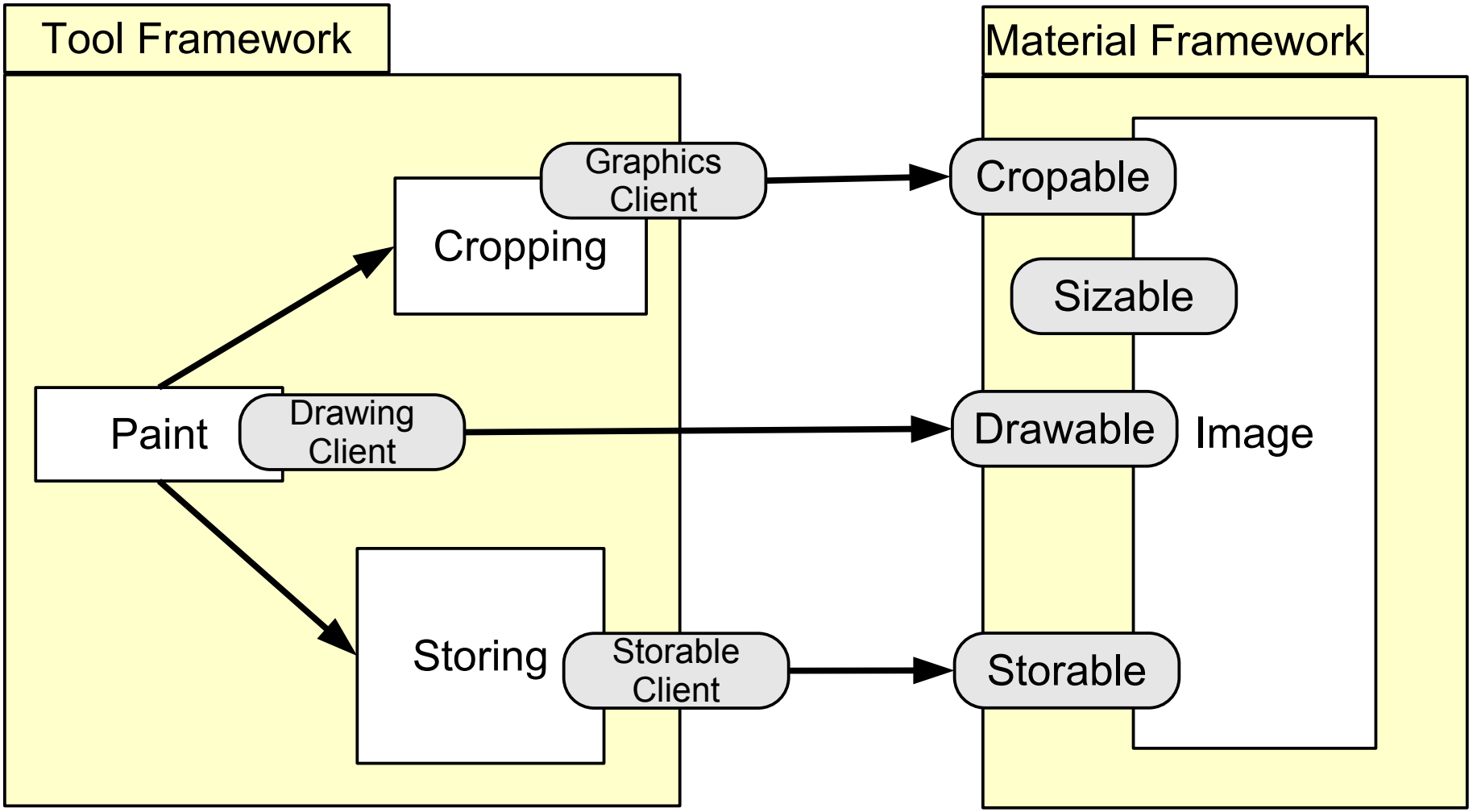
Ex.: Access To Materials In Paint

- Access from tools to material via material-roles
  - Main tool Paint: Drawable
  - Tool Cropping: Cropable via Sizable
  - Tool Saving: Storable



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)

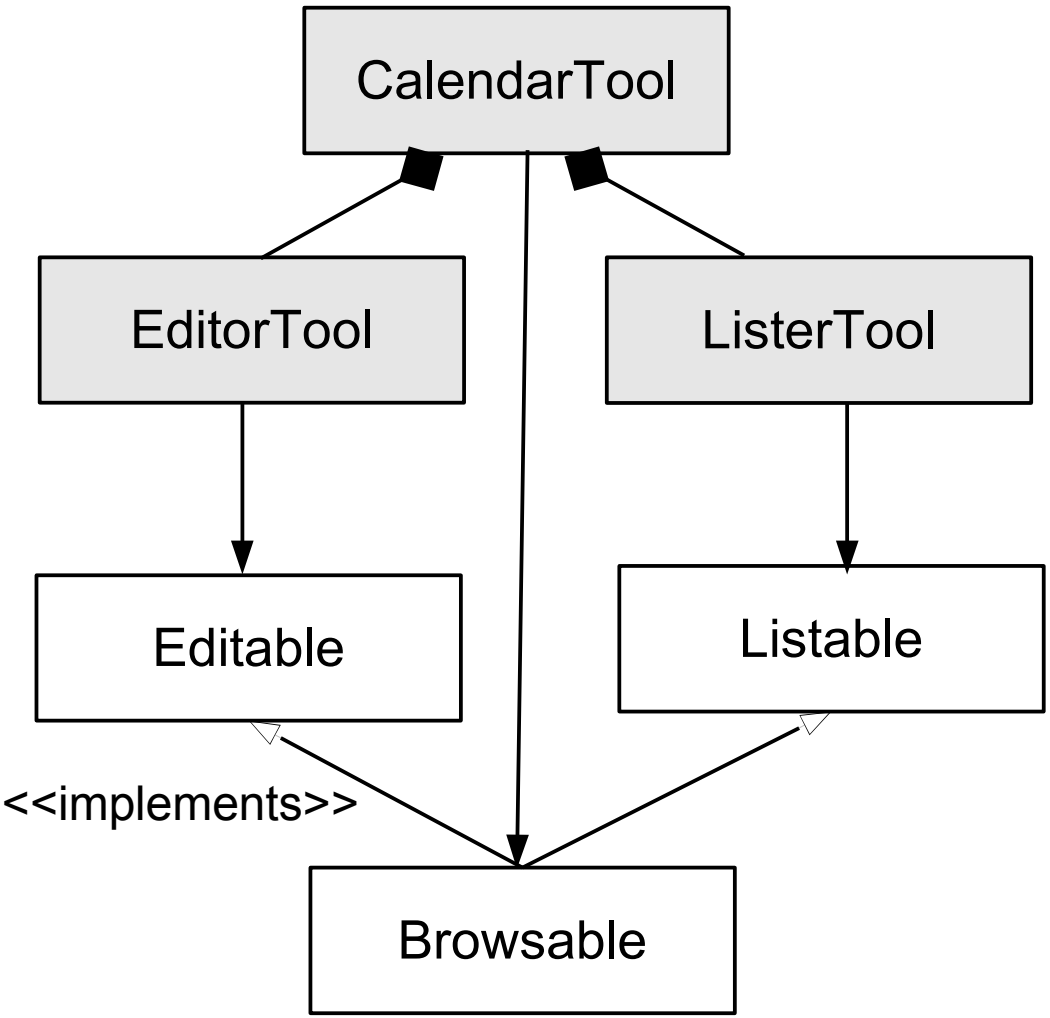


Tool Construction: Structured Tool Pattern

- 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

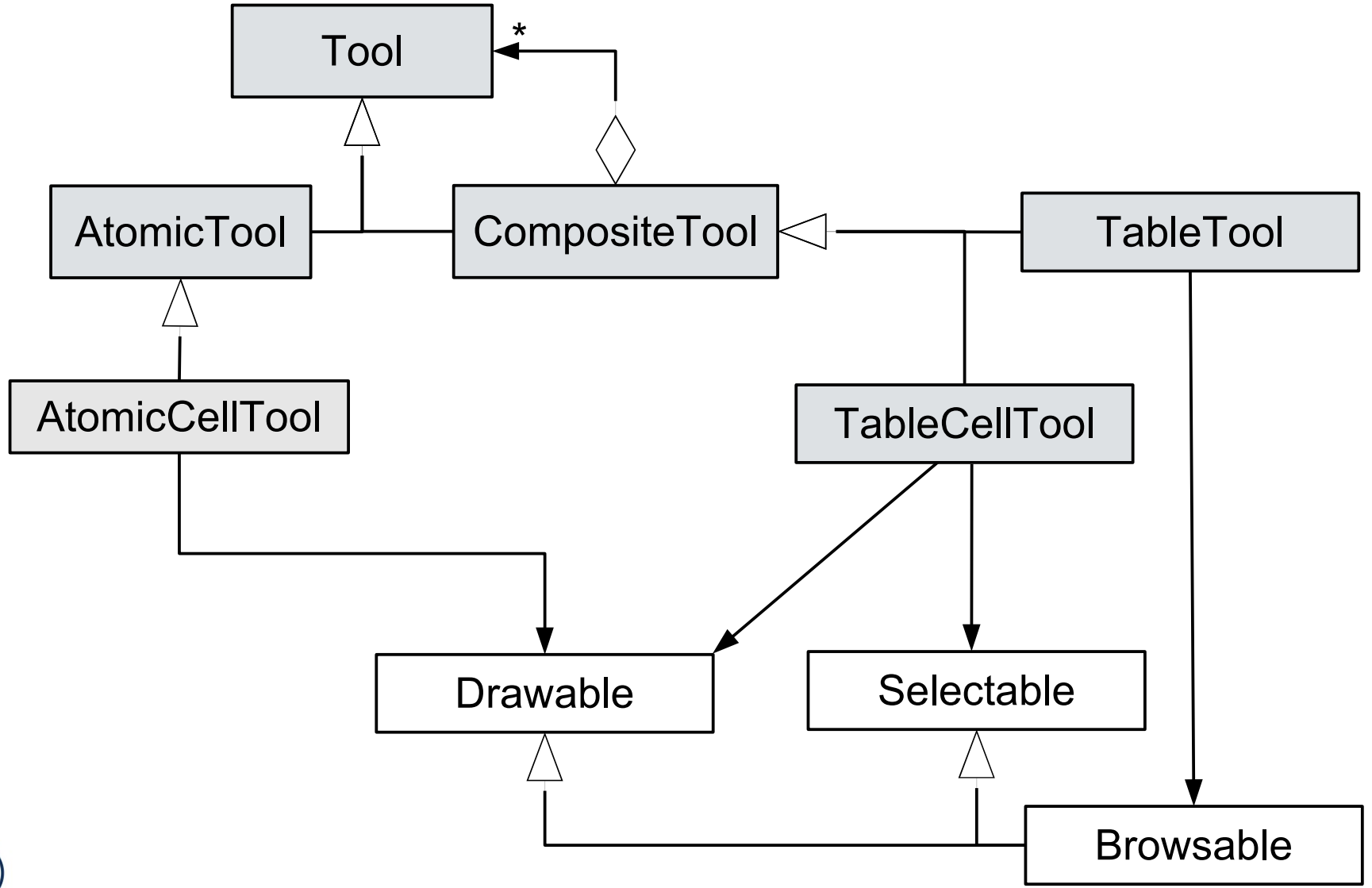
Tool Construction: Structured Tool Pattern

- 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)



Tool Construction: Composite as Structured Tool Pattern

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

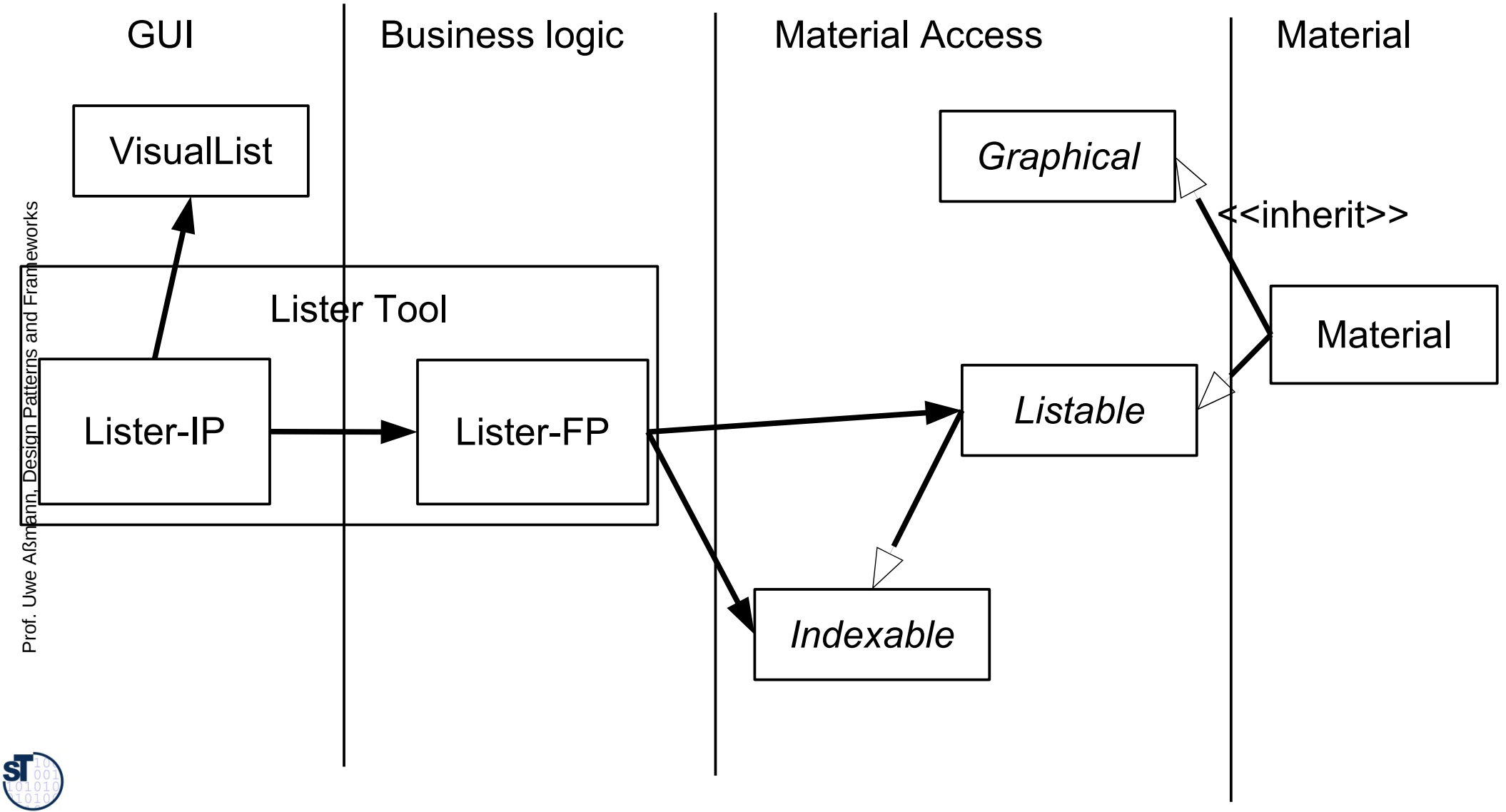


Tool Construction: Separation of Function and Interaction

- 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 parts
- 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

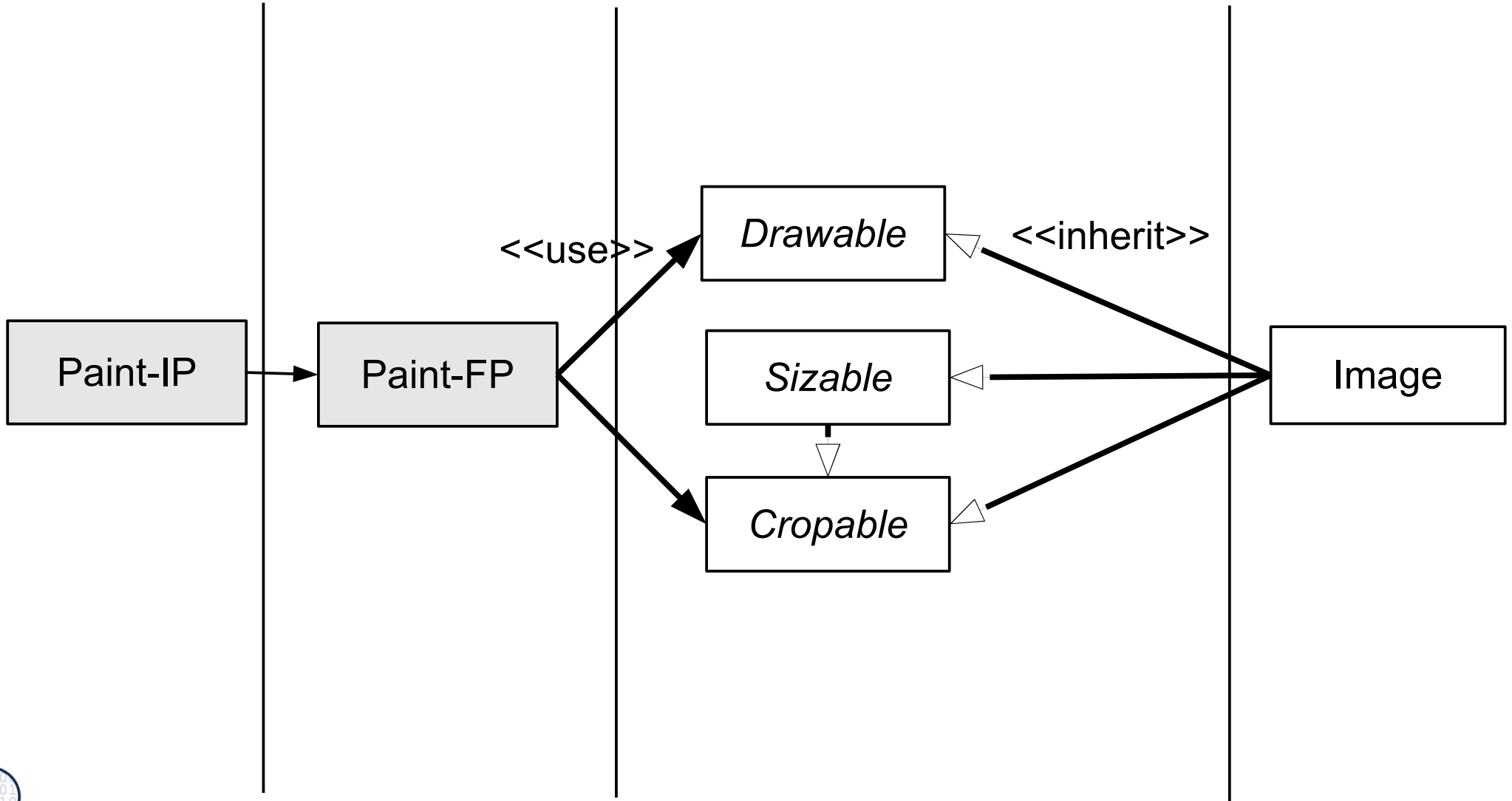
Interaction Part (IP) and Functional Part (FP)

- FP create a new layer



Interaction Part (IP) and Functional Part (FP)

- Paint could be split into IP and FP

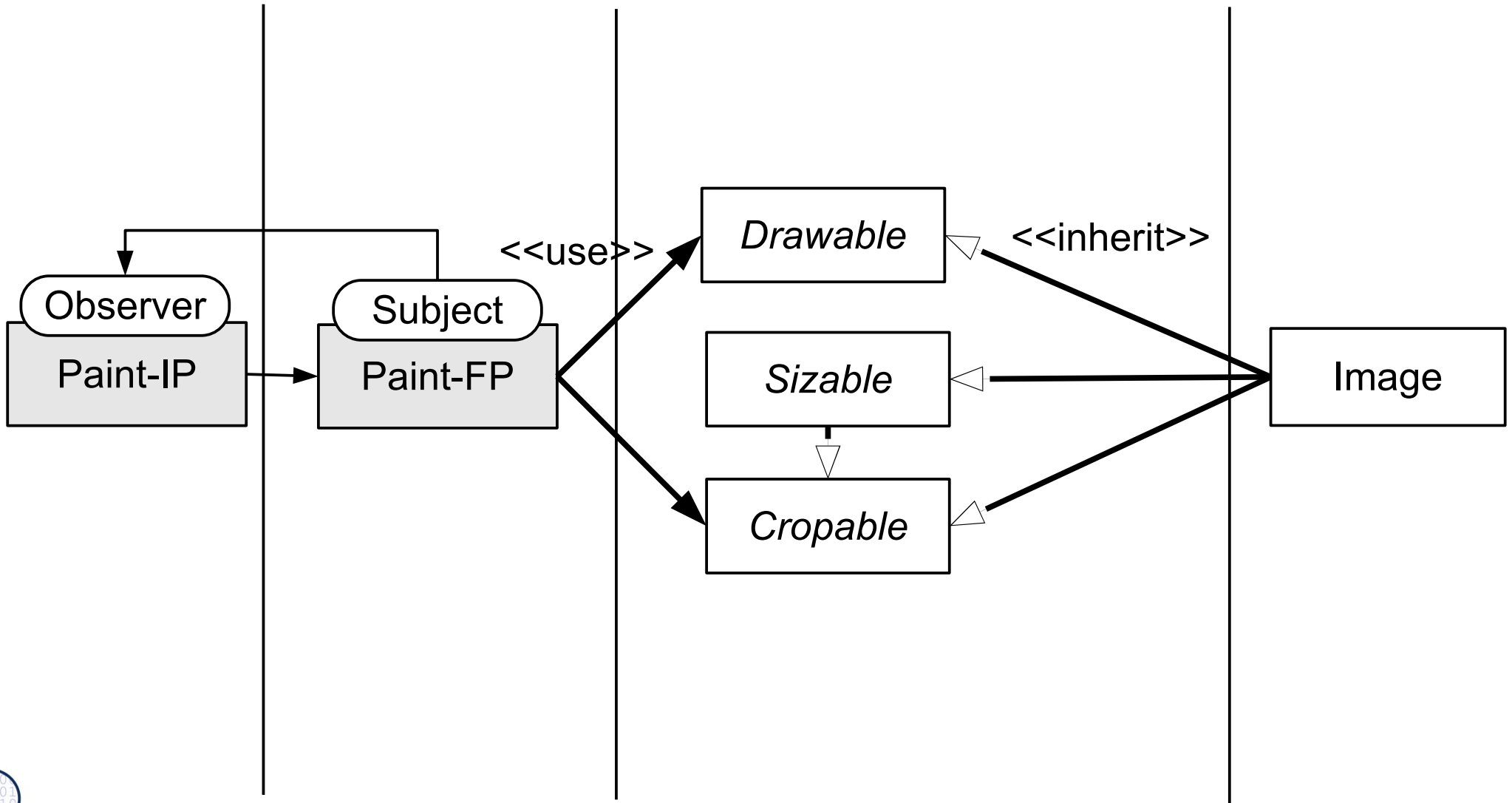


IP-FP TAM Refines MVC

- 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

IP-FP Coupling by Observer

- Paint could be split into IP and FP

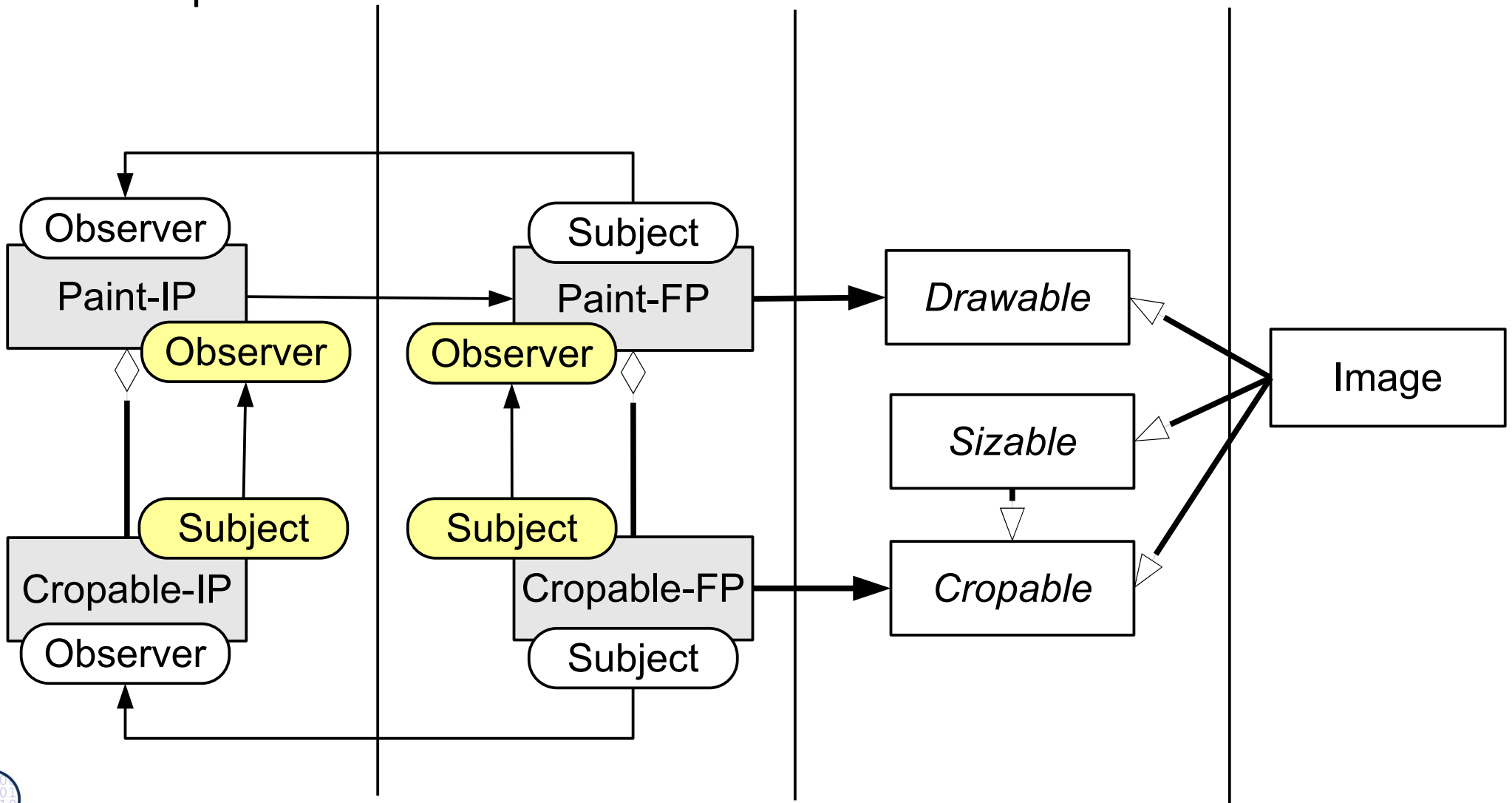


Coupling between Subtool-FP and Supertool-FP

- **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)

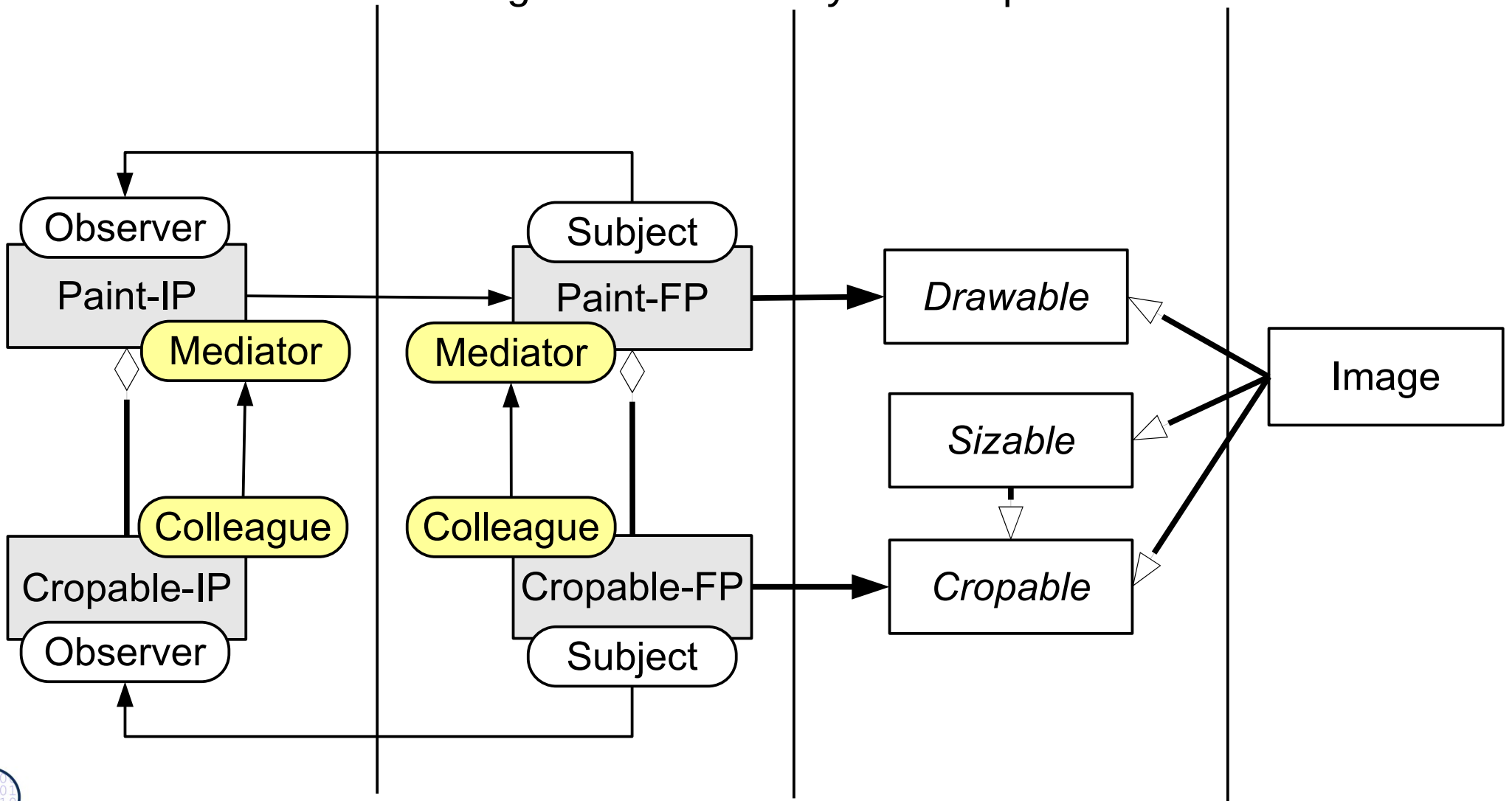
Subtool and IP-FP Coupling by Observer

- IPs observe FPs
- Supertools observe subtools



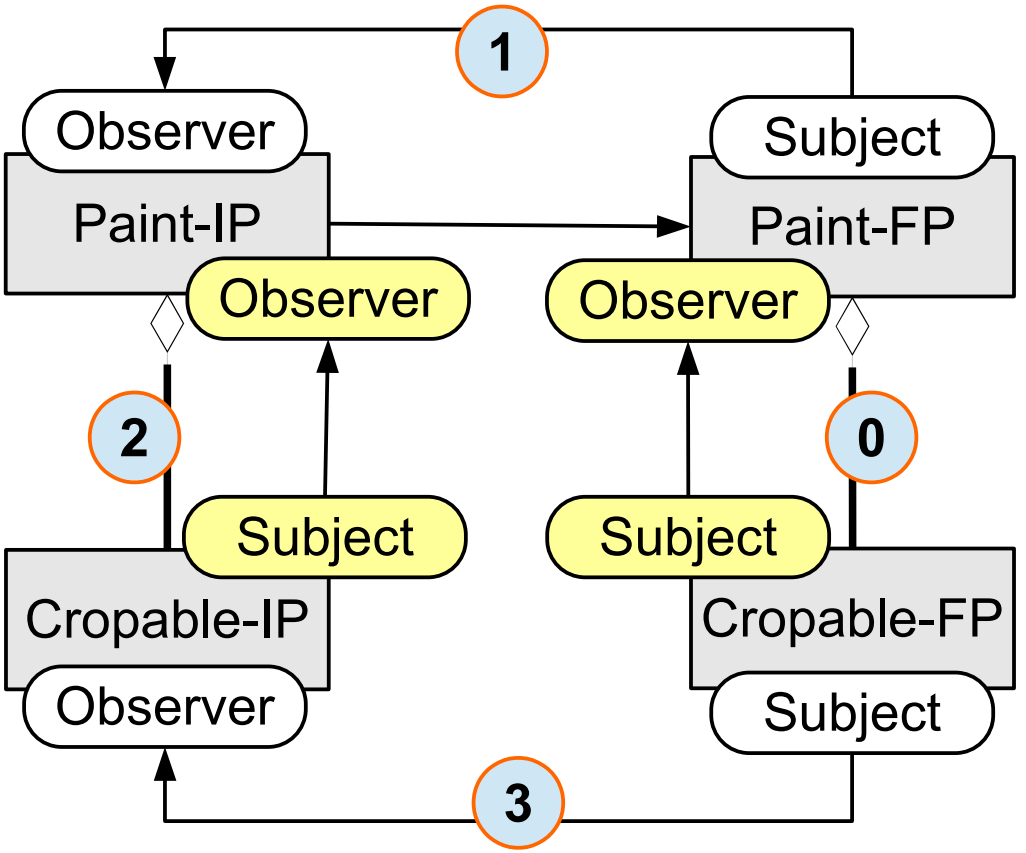
Subtool Coupling by Mediator

- IPs observe FPs
- Subtools are colleagues mediated by their supertool

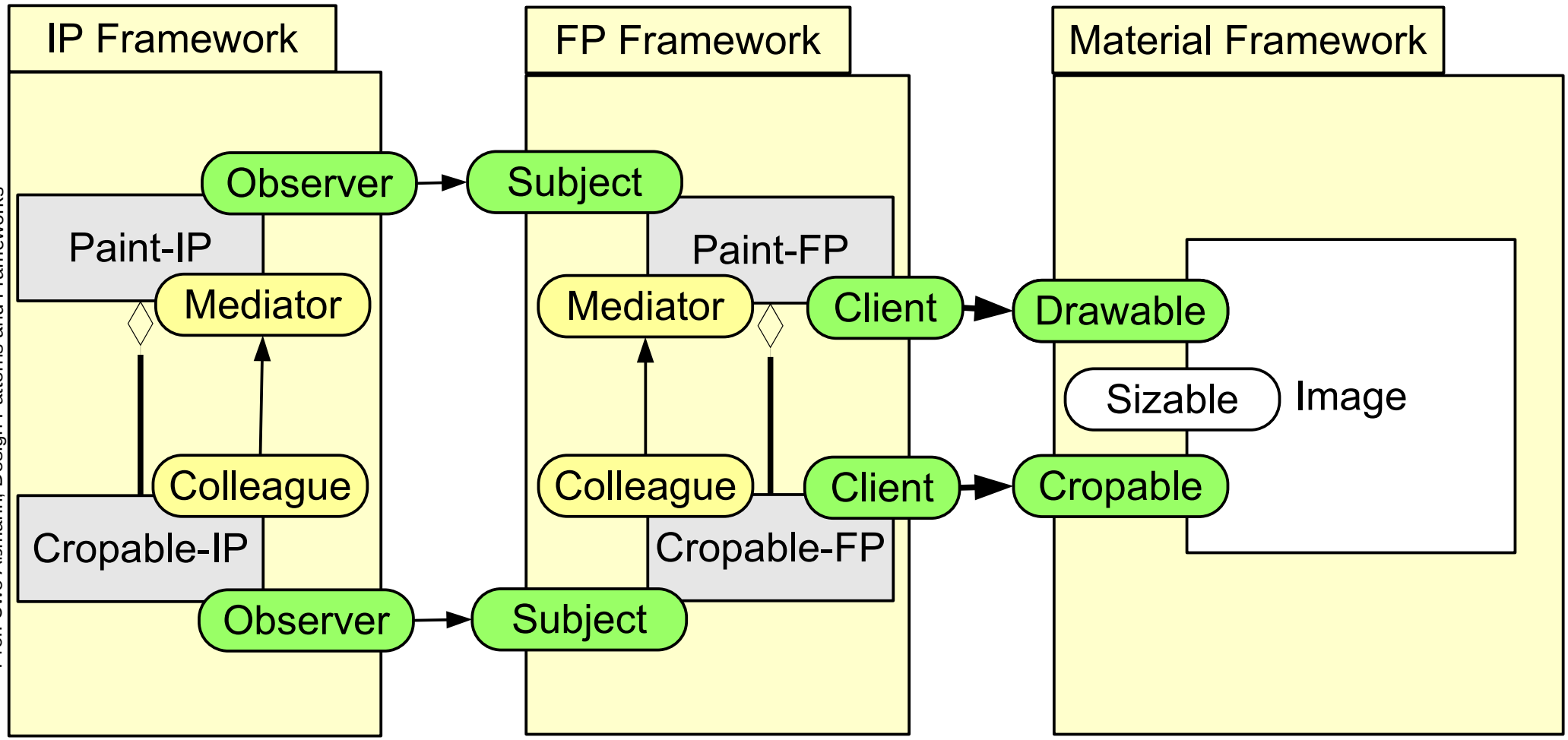


Creation of New Subtools

- 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



Paint in Framework Notation

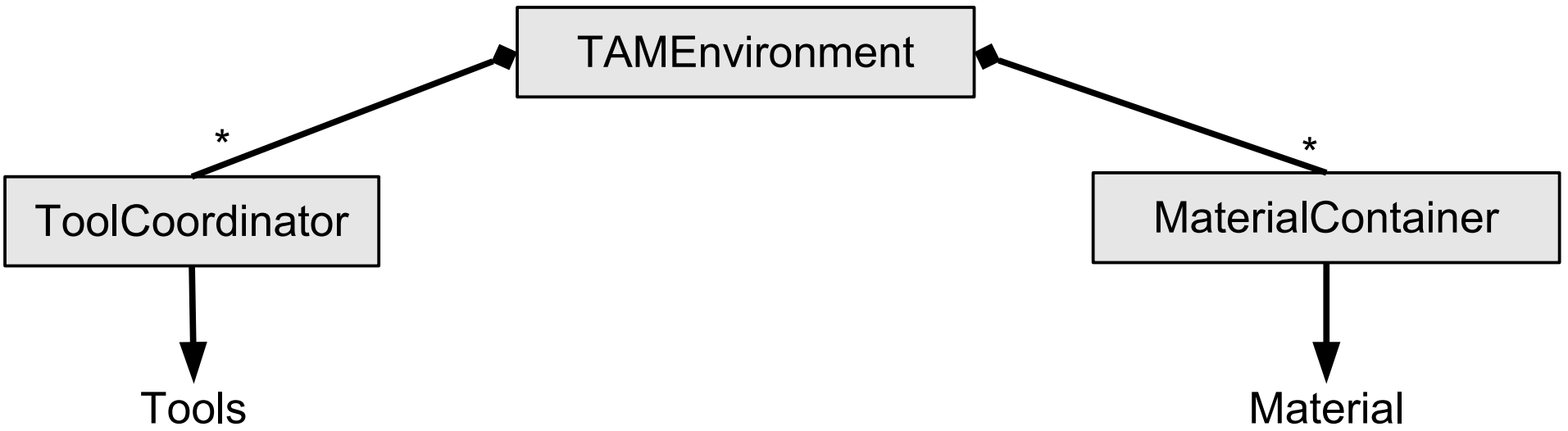


14.3 TAM Environment

The Environment

- Tools and Materials live in an environment with
  - Tool coordinators
  - Material container

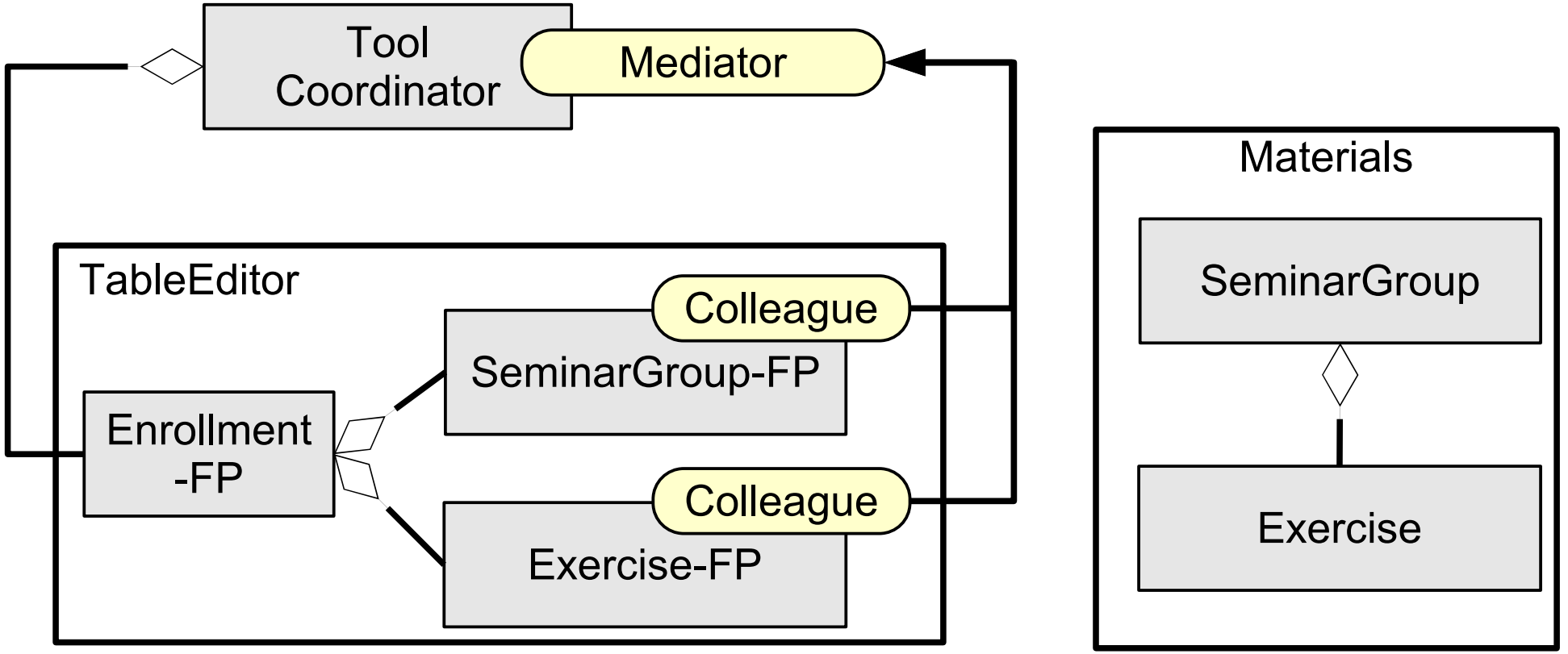
- The environment initializes everything, displays everything on the desktop, and waits for tool launch



Tool Coordinator in the Tool Environment

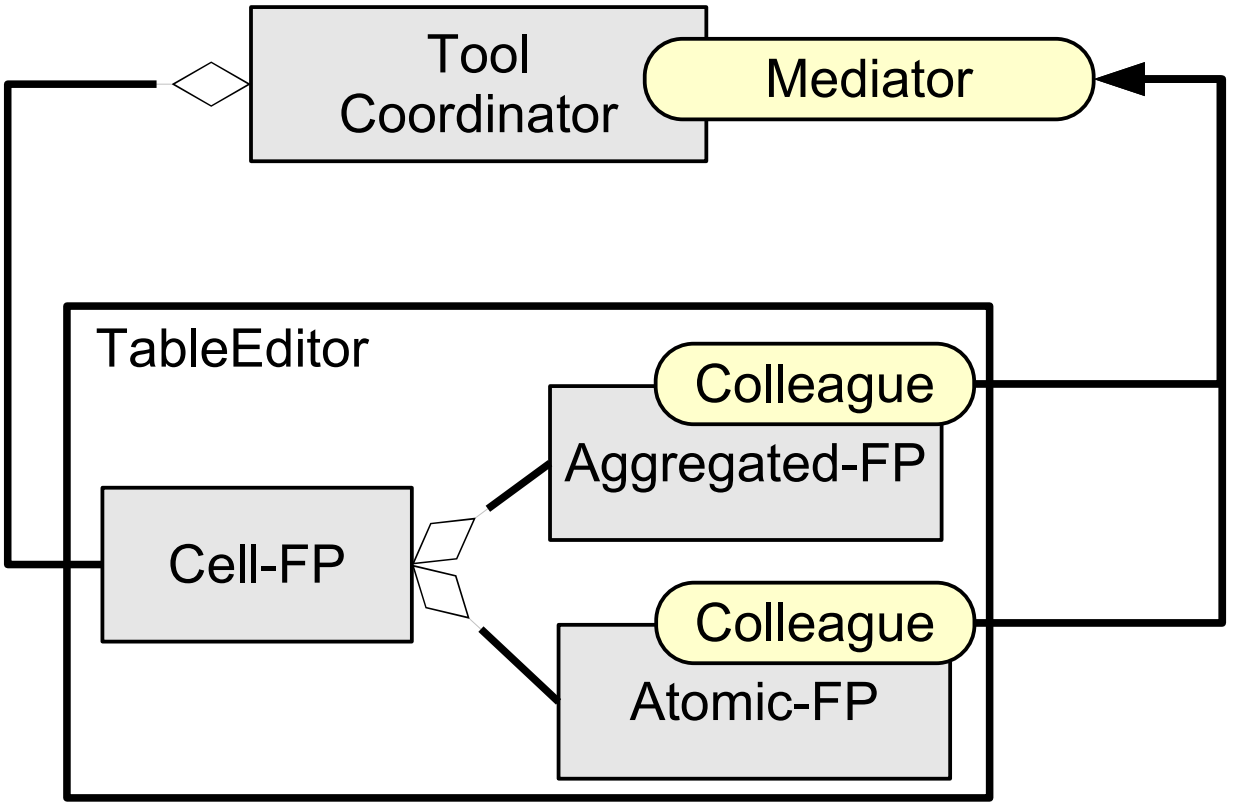
- The **Tool Coordinator** is a global object
  - Groups a set of tools and their related material
  - Contains:
    - A Tool-Material dictionary of all tools and the materials they work on
    - A tool factory
- Is a Mediator between FPs and other tools
  - Usually, FPs talk to their supertools and their related IPs. When materials depend on other materials, other tools have to be informed
    - Examples:
      - aggregation cell in a table,
      - enrollment conditions for an exercise part of a seminar group
  - The ToolCoordinator uses the Tool-Material dictionary to notify tools appropriately

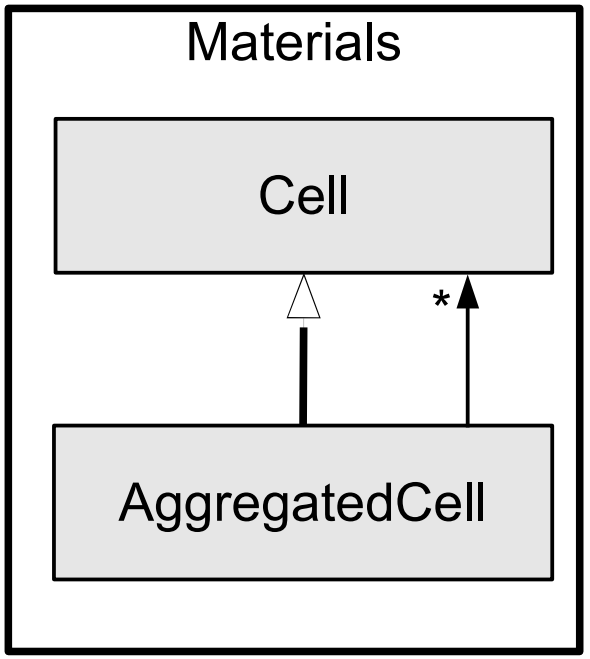
Example: Seminar Groups

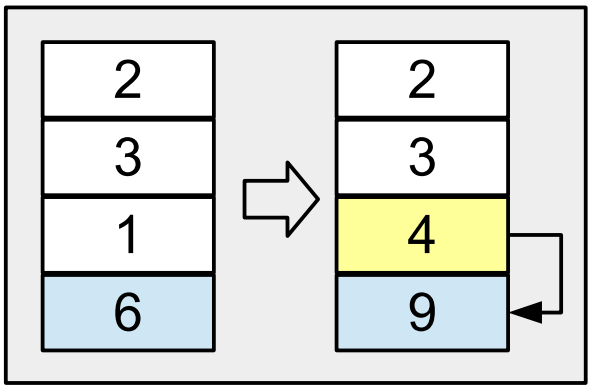


- A seminar group for 30 students should only comprise exercises which allow for at least 30 students to enroll
- Updating an exercise, which is part of a seminar group, requires to check this constraint on the containing seminar group

Example: Aggregation Cell





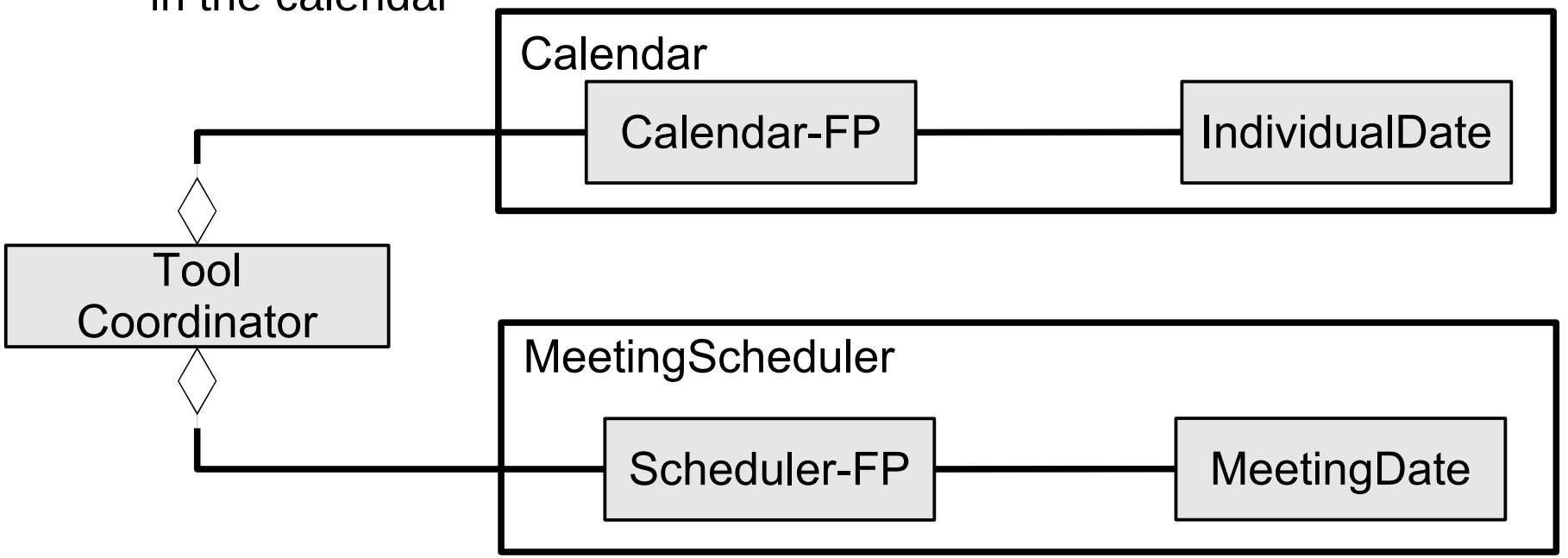


- The Cell-FP has to remember which cells are referenced by aggregation cells
- This aspect is extracted to the tool coordinator

14.3.1. Pattern: Constrained Material Container

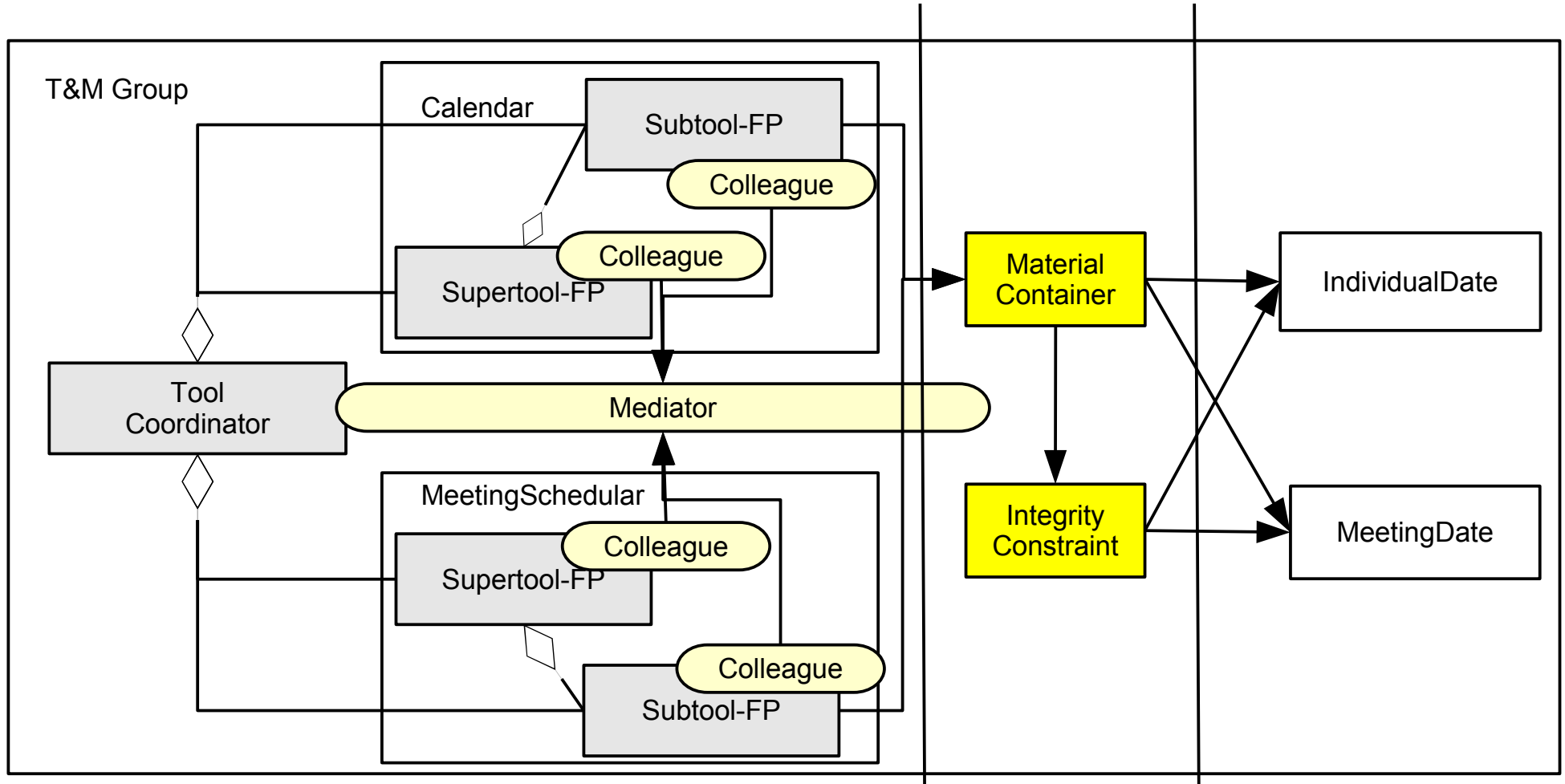
Problem: Dependencies Among Materials

- Materials may depend on each other, i.e., have a semantic overlap
- Example MeetingScheduler
  - Maintains regular meeting dates (week, month, year)
  - Should collaborate with the Calendar tool that maintains individual dates
- Clearly, these materials depend 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 materials that depend on each other into one *Material container*
  - And associate a *constraint object* that maintains the dependencies
  - This way the container encapsulated the (read/write) access restrictions to materials



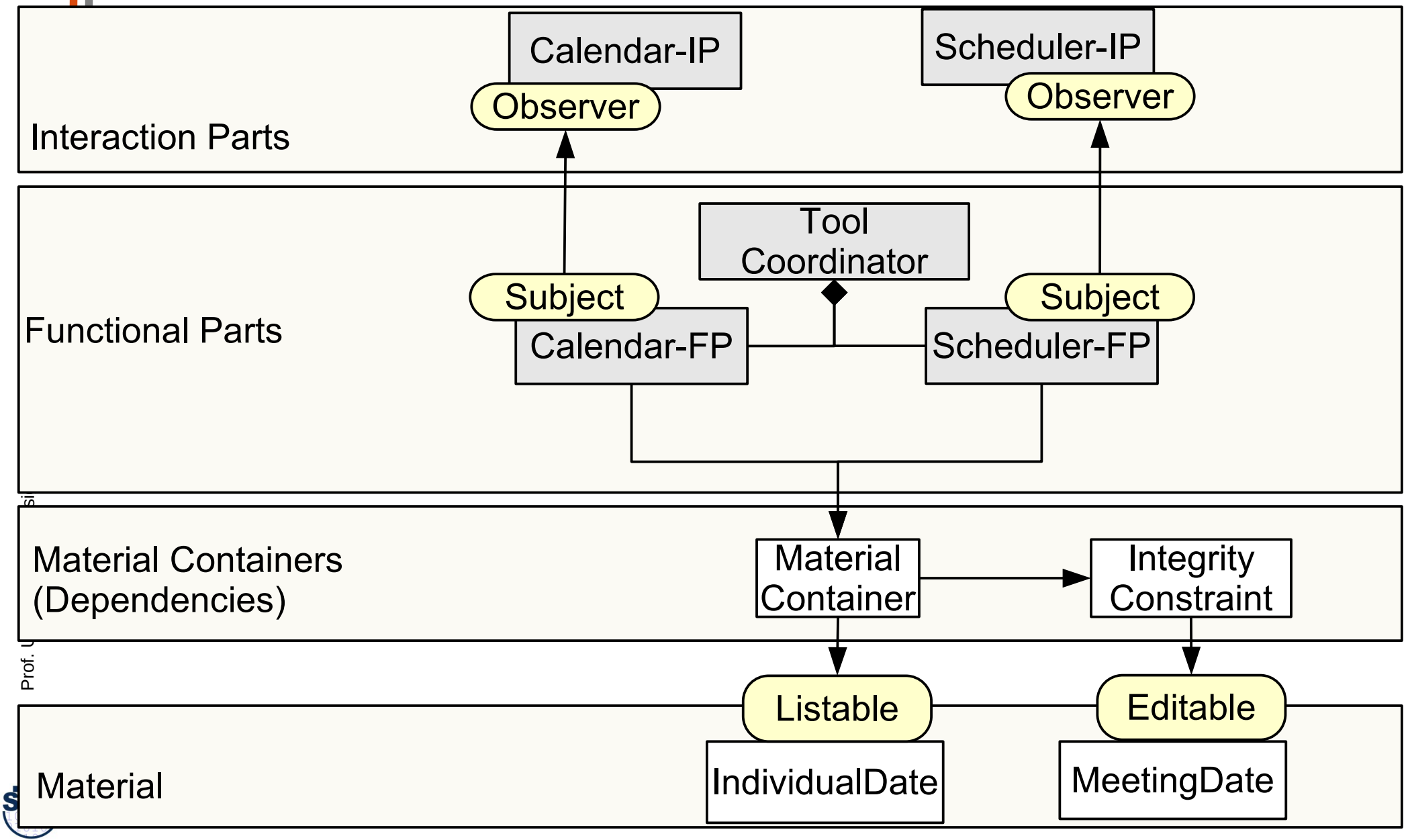
Tool Coordinator and Material Container

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

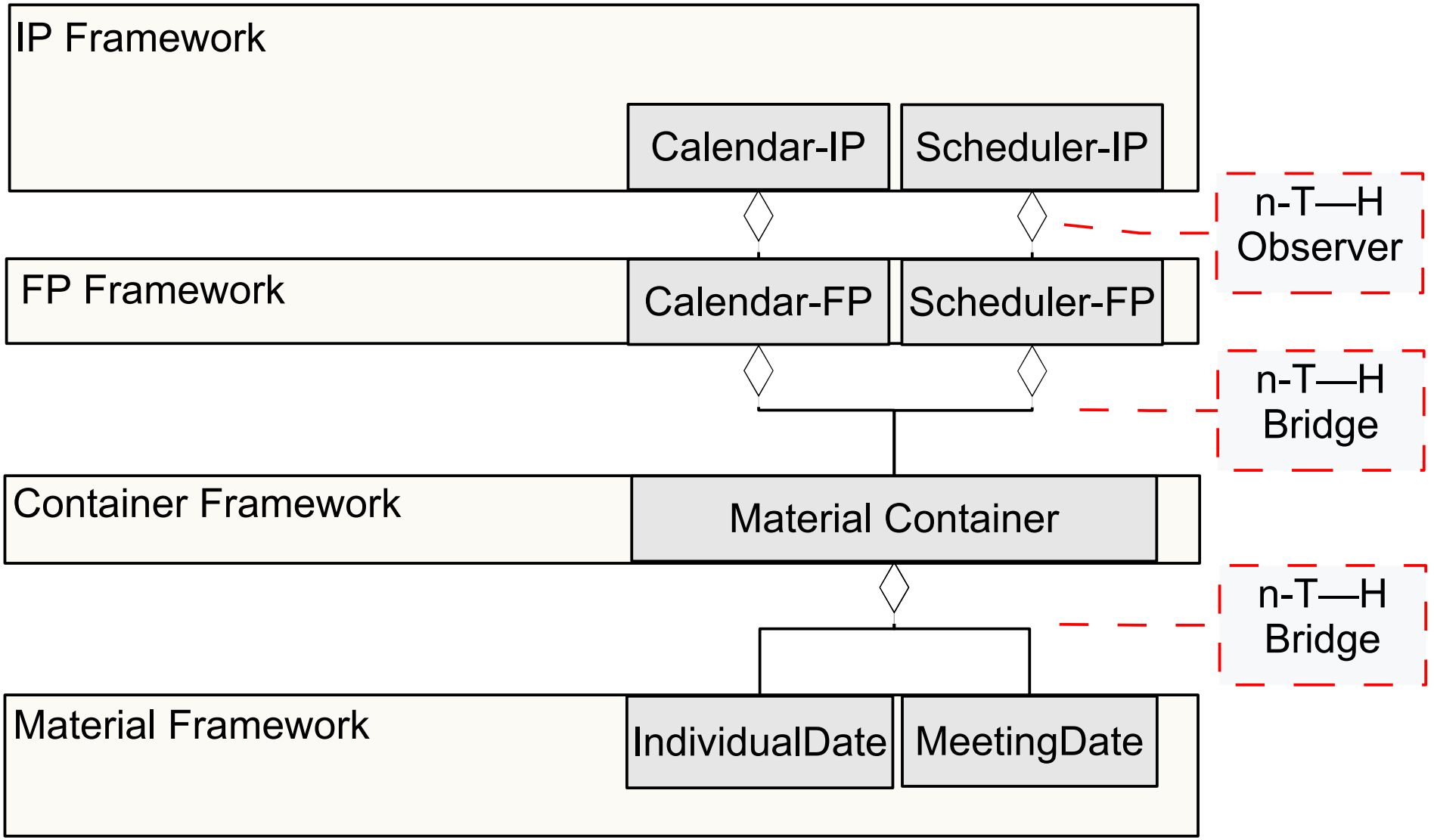
14.4 TAM and Layered Frameworks

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

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 Material Container
  - n-T—H Bridge between Material Container 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

Summary

- The T&M conceptual pattern is a very important pattern for object-oriented development
  - Active tools
  - Passive materials
  - Separation of IP and FP
  - (Work)Environment with
    - Tool Coordinator
    - Material Container
- T&M is a pattern language for constructing interactive applications
  - Refines 3-tier and MVC
  - Uses Command, Strategy, Observer, Composite, etc.
- TAM is a variant of a layered framework, using n-T—H miniconnectors (Observer, Bridge) between the layers

The End