DPF EXCERCISE #11

TUD, ST Group
Dr. Sebastian Götz

Tools and Materials

Basic Tools and Materials (TAM)

1a) Leitmotif, Usage Model & Design Metaphor

1b) Tool Material Interaction

1c) Tool Consituents

1d) TAM Application Layers

Literature

- Dirk Riehle and Heinz Züllighoven:
  A pattern language for tool construction and integration based on the tools and materials metaphor.
  In Proceedings of Pattern languages of program design, 1995.

- http://www.itu.dk/courses/SU/E2004/15-riehle-zuelligh.pdf

Task #1 – Terminology

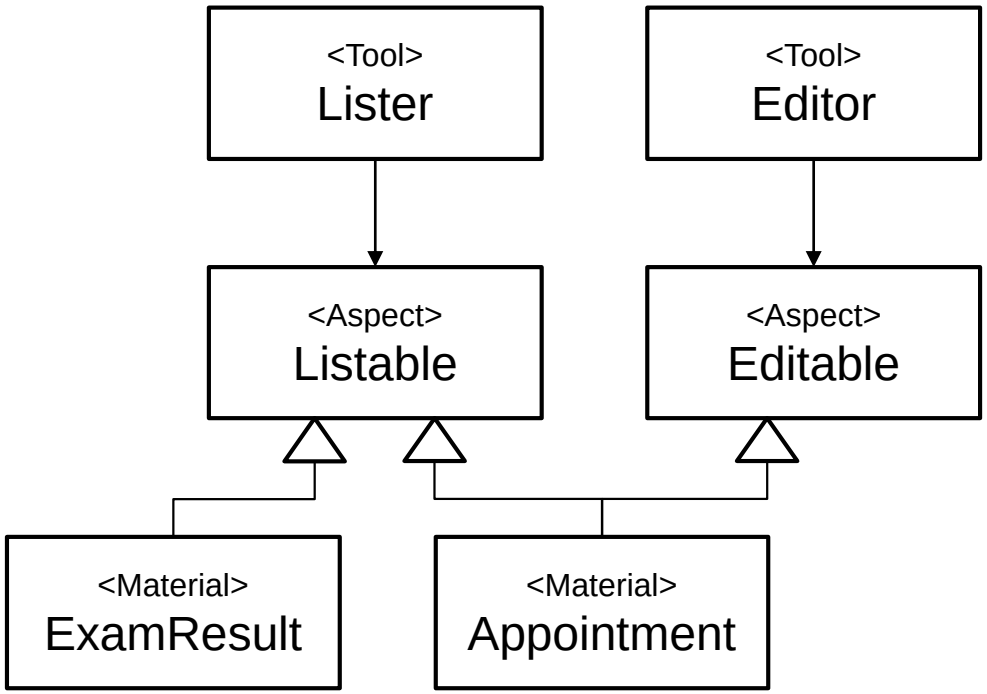
- Usage model
  - a domain-oriented model of how the software can be used to perform activities in its usage context.
- Leitmotif
  - a common frame of orientation for the parties concerned in development and usage of the application.
  - supports design, usage, and evaluation of software and is based on values and objectives (e.g., scientists, patriots, optimists, …).
- Design metaphor
  - a figurative concept used to make a leitmotif more concrete (i.e., towards implementation and design patterns)

It is sensible to use **design metaphors** based on a **Leitmotif** to realise a **usage model**.

Task #1 – Terminology

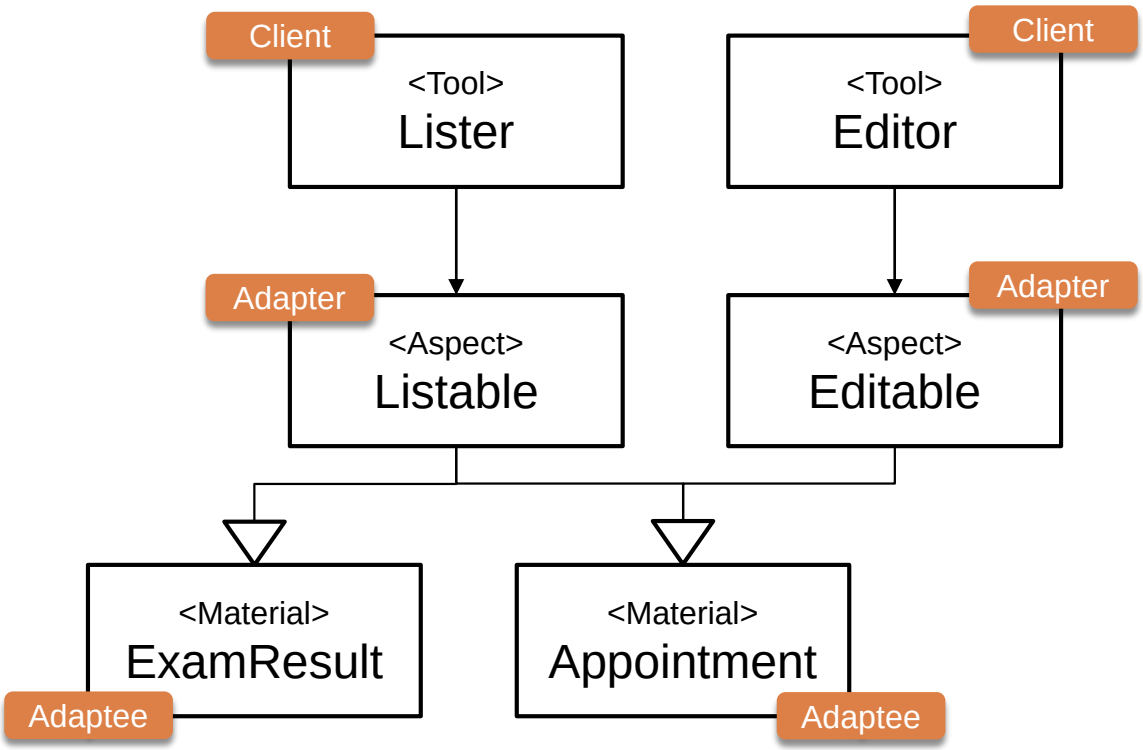
- Leitmotif of TAM
  - Independent experts
    - Objective: supportive & flexible system
    - Values: detailed knowledge, high capabilitise
- Design Metaphors in TAM
  - Material ← manipulated by an expert
  - Tool ← enabler for expert to perform manipulation
  - Work environment

Task #1 – Tool Material Interaction



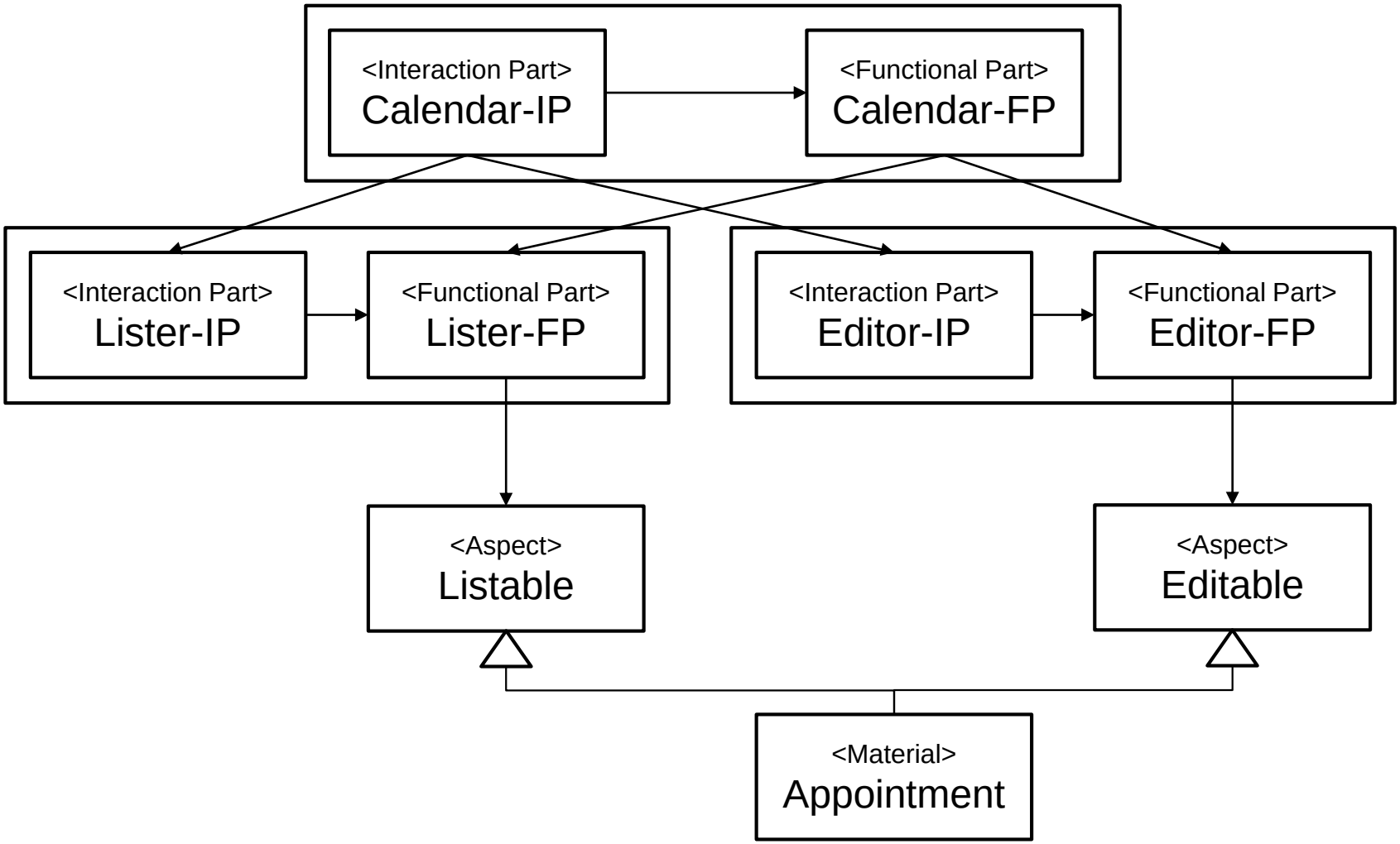
- Users want to be able to *manipulate* a **material** with different **tools**.
- The same tool may be used for different kinds of materials. Tools work on aspects, which are implemented by materials.
- This can, for example, be implemented by using Adapter, Decorator, or Extension Object.

Task #1 – Tool Material Interaction

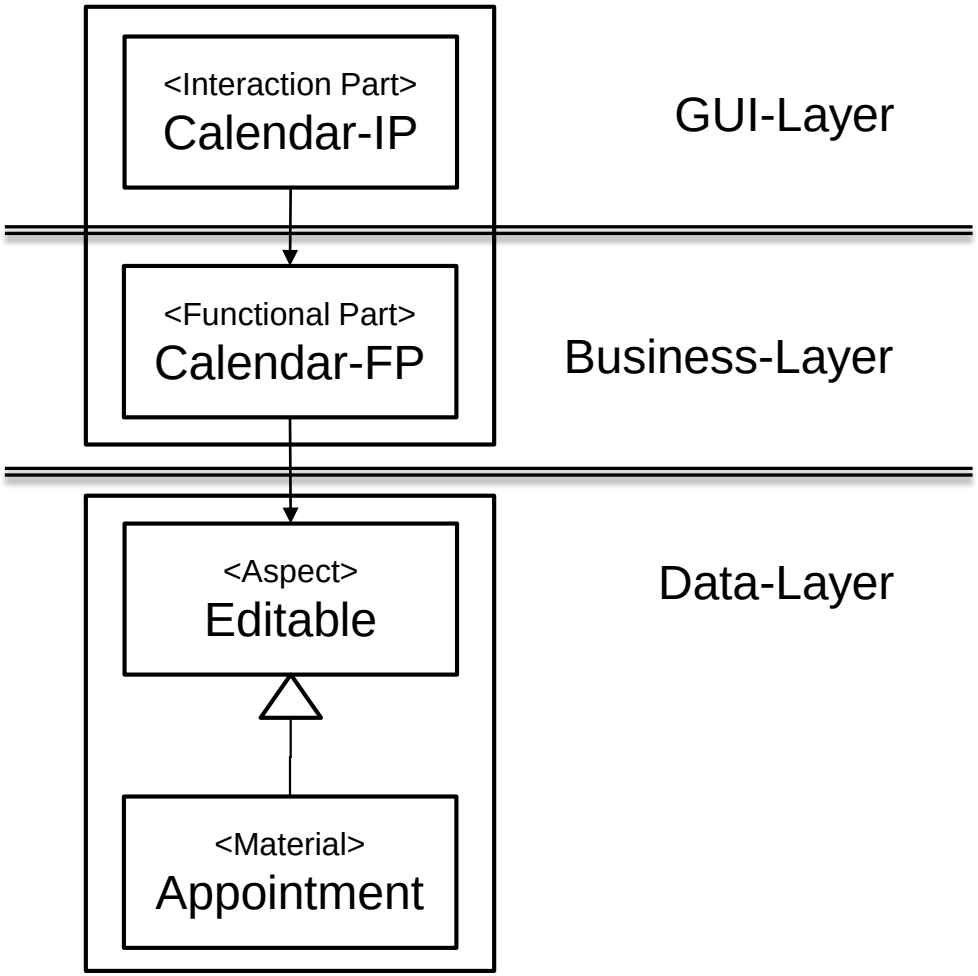


- Users want to be able to *manipulate* a **material** with different **tools**.
- The same tool may be used for different kinds of materials. Tools work on aspects, which are implemented by materials.
- This can, for example, be implemented by using Adapter, Decorator, or Extension Object.

Task #1 – Tool Consituents



Task #1 – Application Layers



Financial Consultancy Tools

2a) Materials

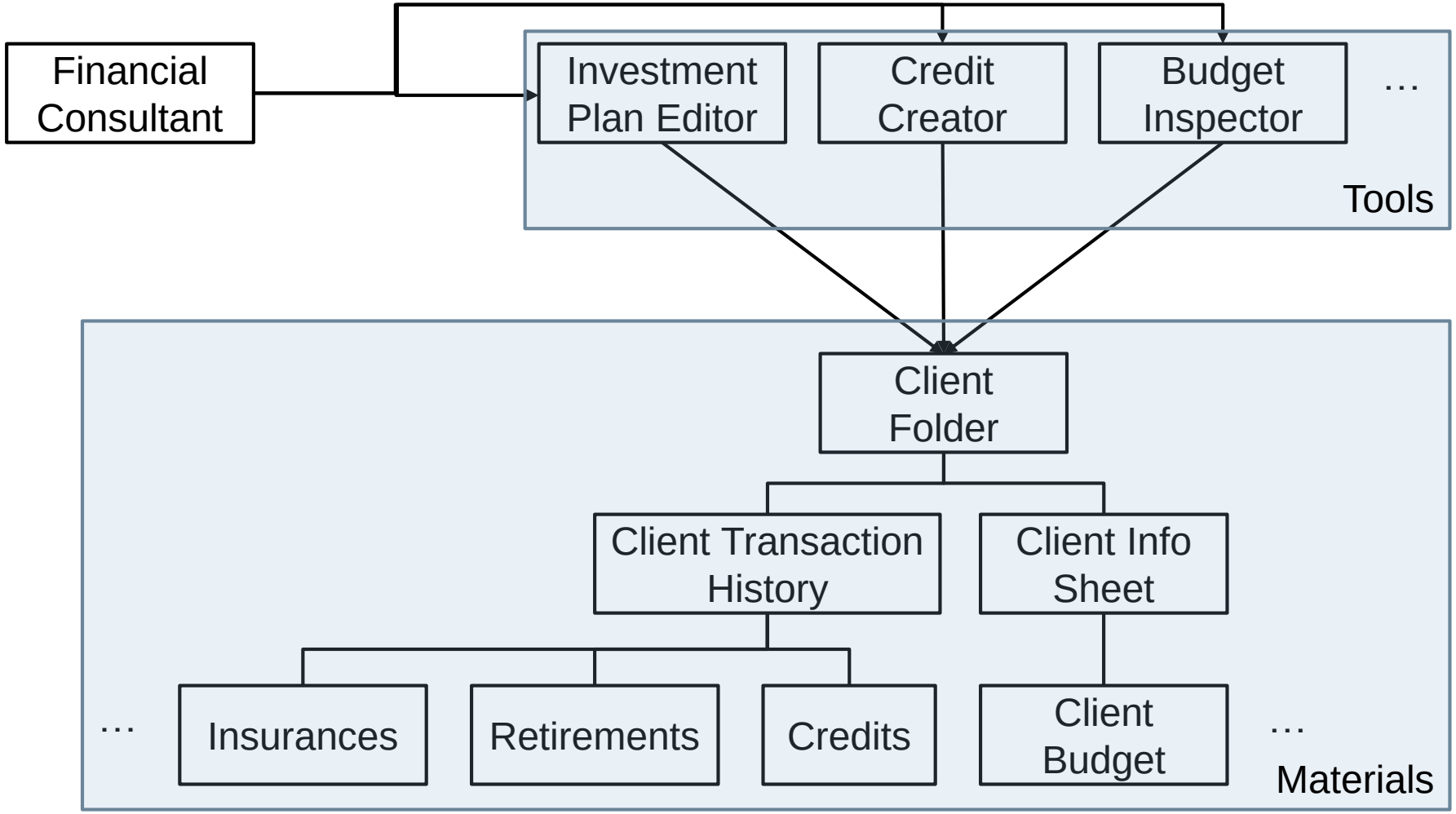
2b) Tools

2c) Benefits

Financial Consultancy Tools

- Financial consultants need to perform a variety of tasks.
  - plan investments,
  - set up credit plans,
  - sell retirement plans etc.
- They maintain a client folder for each client, which contains records of all transactions performed for this client.
  - For example, there is a credit folder which contains data on all credits and loans handed out to this client, and informs about their status.
  - There is also a sheet containing information about the current income and regular costs of the client; this defines the client's monthly budget.
  - There is also information on all insurances the client has obtained.
- When the financial consultant wants to plan a new credit for a client, he must understand the monthly income and obligations of the client. He may also need to analyze the client's investment plan, potentially adjusting it to make room for the credit back payment.
- Using the tools and materials approach, design an application that will support the financial consultant's work.

Financial Consultancy Tools



Benefit of TAM

- Standard development processes use use cases and associated scenarios as a basis for software design.
- The danger is here that **only the exact scenario is supported**, but small deviations—which happen in everyday life—are prevented by the software.
- This is the typical “Sorry, the computer cannot do this” syndrome, where using the computer actually **disallows things that used to be possible in pre-computer times**.

- In the tools and materials approach, the standard scenario is also supported, but because it has been divided among several tools, these tools can also be recombined to **allow for some deviation from the standard scenario**. It is of course still the responsibility of the software developer to make the right decisions when designing the individual tools.

End