14. 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
15-1.1, 1/12/16

This pattern language is one basic strand in the course “Model-Driven Software Development in Technical Spaces” (Winter Term). Welcome!
Literature


- JWAM: Still available on Sourceforge
  http://sourceforge.net/projects/jwamtoolconstr/
  - A copy of jwam.org is in the Internet Archive, also literature
  - Thanks to Moritz Bartl!
Secondary Literature

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
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
TAM vs. 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.

![Diagram of TAM vs. 3-Tier Architectures]

- 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 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 entities
  - Tools are means of work. They embody the experience of how to efficiently work with MATERIAL.
  - Tools present a view on the material.
  - Tools often visible on the desktop as menu entries, wizards, active forms,..
  - Tools interact and give feedback to the user
  - Tools have a state

- If well-designed, tools are transparent, light-weight, and orthogonal
  - However, they should not disappear, since users need to look at a tool if they are worried

- Examples:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Material</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>Contents of a folder</td>
</tr>
<tr>
<td>Interpreter</td>
<td>Code and data</td>
</tr>
<tr>
<td>Calendar</td>
<td>Calendar data, dates</td>
</tr>
<tr>
<td>Form editor</td>
<td>Form</td>
</tr>
</tbody>
</table>
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 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
- We work with different tools on the same material
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

- Requirements Analysis Tool for Task-oriented Requirements Analysis (TORA)
- Editor SANE for activity nets in requirements analysis. 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)
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
    - 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
Tools and Their Views on Material

- Tool → Material Client → <<use>> → ..able Role → Material
- Tool → Material Client → <<use>> → ..able Role
- Tool → Material Client → ..able Role
- Tool → Material Client → ..able Role
- Tool → Material Client → ..able Role
- Tool → Material Client → ..able Role

Prof. Uwe Asmann, Design Patterns and Frameworks
Implementing Tool-Material Roles With ..able-Interfaces
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
Ex.: Access To Materials In TORA

- Access from tools to material via material-roles
  - Main tool *Sane*: Storable
  - Tool *Canvas*: Drawable, Sizable with the help of wrappers DragWrapper, ResizeWrapper
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.: Tools Accessing Material Via Decorators

- Converting roles into decorator objects
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
Since Material-roles are roles, Tool layer and Material layer can be modeled as frameworks (which then can be composed by role composition/use)
10.2.1 Technical Tools vs. Application Tools

- **Technical tools** are stereotypical (interactive) user tasks that reoccur in many applications
  - InteractiveEditor
  - InteractiveLister
  - Loader
  - QueryEngine
  - Inspector
  - Compressor
  - Encryptor

- **Application tools** are specific to an application and carry application-specific semantics
  - Datebooking
  - Mail-sender
  - Page-renderer
  - Slide-presenter

- The Quasar reuse law of [Siedersleben] structures all components into Technical and Application components
Exc.: Identify Tools and Materials

- Open your favorite mail tool and investigate its tools and materials.
  - Which menu items refer to technical tools?
  - Which menu items are application-specific tools?
  - What is the material of the mail tool?
  - Which application-specific tool do you miss?
10.2.2 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
  └── AtomicTool
      └── AtomicCellTool
          └── Listable
          └── Drawable
          └── Zoomable
          └── Selectable
             └── Browsable

  └── CompositeTool
      └──_TABLETool
          └──_TableCellTool
              └── Listable
              └── Drawable
              └── Zoomable
              └── Selectable
     └── Browsable
```
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):** Controller and active part of application
  - Manipulation of the material
  - Access to Material via material-roles

- **Interaction Part (IP):** View
  - Reactive on user inputs
  - Modeless, if possible
  - Can be replaced without affecting the functional part

- IP-FP TAM Refines MVC
  - The “application model” is split between tool-FP, material access, and material
14.3 Interactive and Functional Parts of Tools
Interaction Part (IP) and Functional Part (FP)

- FP create a new layer
Example TORA: How TORA Tools Access Their Material

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

- Canvas-FP is split into IP and FP
- Sane-IP
- Canvas-IP
- SaneFP
- Graphical
  - Sizable
  - Drawable
- Composite
- GraphObj
Example Paint Tool: Interaction Part (IP) and Functional Part (FP)

- Paint could be split into IP and FP

Diagram:

- Paint-IP
- Paint-FP
- Drawable
- Sizable
- Cropable
- Image

Relationships:
- <<use>>
- <<inherit>>
14.3.2 Communication of IP and FP

a) IP-FP Coupling by Observer

- Paint could be split into IP and FP which are coupled by Pattern Observer: Paint-IP plays a view of subject Paint-FP
Coupling between Function and Interaction of Super- and Subtools 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 structural patterns such as Bridge, Composite, Decorator, 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)
Example TORA: Symmetric Coupling between Subtools and Supertools by Observer

- Vertical Observer: Supertools are notified from subtools if something changes.
Example Paint Tool: Subtool and IP-FP Coupling by Observer

- IPs observe FPs
- Supertools observe subtools
14.3.2.b) Subtool Coupling by Mediator

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

Diagram:
- Observer
- Subject
- Mediator
- Colleague
- Cropable
- drawable
- sizable
- cropable
- Image
Example TORA: Coupling between Subtools and Supertools By Symmetric Bureaucracy

IP and FP hierarchy can work with a Bureaucracy each.
14.3.3 Creation of New Subtools

- Initiated by a Super-FP, which decides to create a new sub-FP. Steps:
  - 0) (optional) Sub-FP notifies Super-IP
  - 1) Super-FP notifies Super-IP
  - 2) Super-IP may create one or several sub-IP
  - 3) Super-IP connects them as observers to the sub-FP
Example Paint in Framework Notation

IP Framework
- Observer
- Mediator
- Colleague
- Cropable-IP

FP Framework
- Subject
- Mediator
- Colleague
- Cropable-FP

Material Framework
- Drawable
- Sizable
- Image

Cropable-IP

Observer

Client

FP Framework

Subject

Cropable-FP
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
Example: Generic Editor and Lister Framework

- Supertools are notified from subtools if something changes
- Can be used for every editor and lister of material
Supertools are notified from subtools if something changes.
The Generic Editor in Framework Notation
14.4 The TAM Work-Environment
14.4.1 The Environment

- Tools and Materials live in an *TAM work-environment* with
  - Tool coordinators
  - Material administrations
  - Event coordinators

- The environment initializes everything, displays everything on the desktop, and waits for tool launch
14.4.2 Tool Coordinator in the TAM 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

- Materials and IPs do not talk to the ToolCoordinator
Example of Complex Materials: Information System “Seminar Groups”

- **Constraint:** 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 of Complex Materials: Aggregation Cell

- The Cell-FP has to remember which cells are referenced by aggregation cells
- This aspect is extracted to the tool coordinator
14.4.3. 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)
  - Collaborates 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

Diagram:
- Tool Coordinator
  - Calendar
    - Calendar-FP
    - IndividualDate
  - MeetingScheduler
    - Scheduler-FP
    - MeetingDate
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
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 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 Tool Coordinator

Object Lexicon

Editor-FP

Colleague

Lexicon-FP

Mediator

Sane

Colleague

Sane-FP

Colleague

Activity-FP

InSane Constraint

Material Container

Editable

Lexicon Obj

ActivityObj

Storable

GraphObj
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 managed by a MaterialContainer
  - Uses a ConstraintObject InSaneConstraint to make sure 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
14.4.4 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
- Example: Apple automator tool for desktop automation
The Automaton books regular meetings as dates into the calendar.
14.5 The TAM Pattern
Language and Layered Frameworks

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

Interaction Parts
- Calendar-IP
  - Observer
  - Subject
  - Calendar-FP

Functional Parts
- Scheduler-IP
  - Observer
  - Subject
  - Scheduler-FP
  - Tool Coordinator

Material Containers (Dependencies)
- Material Container
  - Integrity Constraint
  - Listable
  - Editable
  - IndividualDate
  - MeetingDate

Material
TAM and Layered Frameworks (with Framework Packaging)

Interaction Parts
- Calendar-IP
  - Observer
  - Subject
- Scheduler-IP
  - Observer
  - Subject

Functional Parts
- Calendar-FP
- Scheduler-FP

Material Containers (Dependencies)
- Material Container1
  - Material Use1
- Material Container2
  - Material Use2

Material
- IndividualDate
- MeetingDate

Editable

Listable
TAM and Layered Frameworks (with Packaging and Mini-Connectors)

- **IP Framework**
  - Calendar-IP
  - Scheduler-IP

- **FP Framework**
  - Calendar-FP
  - Scheduler-FP

- **Container Framework**
  - Material Container

- **Material Framework**
  - IndividualDate
  - MeetingDate

Diagram elements:
- n-T—H Observer
- n-T—H Bridge
- n-T—H Bridge
TAM and Layered Frameworks

Interaction Parts

Functional Parts

Material Containers (Dependencies)

Material
Example TORA: TAM and Layered Frameworks
TAM Creates Layered Frameworks

- 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!
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
Compare the nature of a tool object and a material object
What are value material objects?
How would you structure an FP part of a tool?
Explain tool-material collaboration and how to implement it
Why are materials not independent of each other? How to cure this?
Why are IP and FP layer isomorphic?
How to create new subtools for a supertool?
Give an example for a structured TAM architecture of an interactive application

Some slides are courtesy to Dr. S. Götz