Presentation-abstraction-control Pattern

Mehmet Aksit
CTIT and Department of Computer Science
P.O. Box 217
7500 AE Enschede, The Netherlands
aksit@cs.utwente.nl
trese.cs.utwente.nl/~aksit

Table of contents

- problem
- forces
- solution
- structure
- dynamics
- design
- exercise
- benefits
- limitations
Problem Description

Design of a complex interactive system, using hierarchy of cooperating agents.

Forces

Agents maintain their own data. However, they must effectively cooperate.
Interactive agents provide their own (user) interface.

Systems evolve over time: The representation of agents may change in time. New agents may be introduced;
Solution

• Structure the interactive agents as a tree-like hierarchy of PAC agents;

• There should be one top-agent, several intermediate agents, and even more bottom agents;

Solution (cont’ed)

• Every agent is responsible for a specific aspect and has 3 components: presentation, abstraction and control;
Solution (cont’ed)

- Dependencies among agents are **transitive**: each agent depends on all higher-level agents.

- The top-level agents provides the **functional core** of the system;

Solution (cont’ed)

- Bottom-layer consists of self-contained semantic concepts on which users of the system can act.
Solution (cont’ed)

- Intermediate-layer agents represents combinations/relations between the lower level agents.

Structure of the top-level agent

- **Abstraction component:**
  Offers functions to manipulate the data model and to retrieve information about. Should be media independent.

- **Presentation component:**
  It may include user-interface elements common to all.
Control component of the top-level agent

• Control component:

(i) Allows lower-level agents access the top-level agents;

(ii) It maintains information about the connections;
Control component of the top-level agent

- Control component:
  
  (iii) maintains information about the interactions of the user and keeps logs.

Presentation component of the bottom-level agent

- Presentation component:
  Presents a specific view of the corresponding semantic concept;
**Abstraction component of the bottom-level agent**

- **Abstraction component**: Maintains agent-specific data; No other agents depend on this data.

**Control component of the bottom-level agent**

- **Control component**: Maintains consistency between the abstraction and presentation components avoiding direct dependencies between them;
- **Communicates with higher-level agents to exchange events and data.**
Intermediate component

- **Composition:**
  Intermediate agent groups presentations of each lower-level agent to form a composite graphic object. It defines a new abstraction;

- **Coordination:**
  It maintains consistency between the lower-level agents.

---

**Scenario 1: creating a new bottom-level agent (bar chart)**

- **Top-level**
  - openView(barChart)

- **View coordinator**
  - receiveMsg(open)
  - receiveMsg(getData)

- **Bar chart**
  - control
  - abstraction
  - presentation
  - setChartData
  - open
  - getData
  - getChartData
Scenario 2: entering data

Top-level Scenario dynamics

Spread-sheet View Bar chart

topData enterData receiveMsg(setData)

.setData receiveMsg(change)

receiveMsg(setData)

receiveMsg(change) receiveMsg(getData)

data receiveMsg(change)

getBar receiveMsg(getData)

Design (implementation)

1. Define a model of abstraction;
   Problem?
   Model?

2. Define a general strategy;

3. Specify the top-level agent;

4. Specify the bottom-level agents.
5. Bottom-level agents for system services;

6. Intermediate level agents for composing the lower-level agents;

7. Intermediate level agents for coordinating the lower-level agents;

8. For each agent introduce presentation and abstraction component;
Design (implementation)

9. Provide the external interface and link the agents together;

Exercise: Designing the car editor as a PAC pattern

• Consider the car editor game. Identify the top-level agent.

• Identify the bottom-level agents.

• Identify the intermediate-level agents.
Exercise: component creation in the car editor

- Refer to the first example scenario.
- Select a car component as a low-level agent;
- Define the interaction diagram of creating a new car component.

Exercise: coordinating car components

- Refer to the second example scenario.
- Assume that a car component data has been modified;
- Define the interaction diagram of updating the other related components;
Exercise: Integration of the sub architectures

• How would you integrate the structures defined by the car models and the PAC editor?

Benefits

1. Separation of concerns:

2. Support for change and extension;

Issues and limitations

- Increased complexity;
- Complex control component;
- Efficiency;
- Applicability.