Exercise 2: Implementing Patterns

Issued: **May 10th, 2006**, Return for corrections: **May 23rd, 2006**

MASTER SOLUTION

The Blackboard pattern

1) Learn about the pattern

There are numerous resources on the web. The blackboard pattern seems not as well defined as the patterns in the GoF book. The blackboard pattern can thus be described more as a general solution strategy than a concrete template. Most of the examples of this master solution is based on the pattern description as found in

http://chat.carleton.ca/~narthorn/project/patterns/BlackboardPattern-display.html

2) Find an example use of the pattern

Example uses of the patterns include:

- Mathematical reasoning systems (see below)
- Distributed computation systems (see for example “SETI@home” or “distributed.net”)
- Artificial intelligence systems
- Data networks (skyserver.sdss.org)

The following example illustrates a mathematical reasoning system that calculates the result for formulas of the following form:

$$\frac{a_1}{b_1}+\frac{a_2}{b_2}+\frac{a_3}{b_3}+\ldots+\frac{a_n}{b_n}$$

We assume that we have two different specialists that work on the formula: One specialist knows how to bring two quotients to the same value. The second specialists knows how to simplify the term by adding two fractions.

3) Implementation

There are three distinct roles that participate in the blackboard pattern: BLACKBOARD, SPECIALIST and SUPERVISOR. The BLACKBOARD stores intermediate and final results. The SUPERVISOR controls the actual calculation by scheduling the SPECIALISTs. SPECIALISTs implement the actual functionality by changing the data stored in the BLACKBOARD.

SPECIALISTs register with the BLACKBOARD. They can be queries if they want to update the data stored on the BLACKBOARD. The SUPERVISOR then picks out one of the “ready” SPECIALISTs and lets him update the data.

Blackboard pattern framework (sketch)

```
class BLACKBOARD[G]
feature -- Access
    data: G
        -- Data stored in the blackboard
feature -- Data update
    set_data (v:G)
            -- Update the data to value `v' and notify the specialists.
        ensure
            data_set: data = v
feature -- Handle specialists
    subscribed_specialists: SET[SPECIALIST[G]]
    subscribe (a_specialist: SPECIALIST[G])
            -- Subscribe `a_specialist' to this blackboard.
        ensure
            subscribed: subscribed_specialists.has (a_specialist)
end

deferred class SPECIALIST[G]
feature -- Status report
    has_update: BOOLEAN
        -- Is an update available?
feature -- Access
    blackboard: BLACKBOARD[G]
        -- Blackboard the specialist is subscribed to
feature -- Notification
    check_for_update is
            -- The data on the blackboard has changed, see if there is anything new
            -- to contribute.
        deferred
        end
    update_blackboard is
            -- Update the blackboard.
        require
            update_available: has_update
end

class SUPERVISOR[G]
feature -- Access
    blackboard: BLACKBOARD[G]
        -- Supervised blackboard
    best_specialist: SPECIALIST[G]
            -- Best specialist (heuristic), computed from the current set of ready specialists
        ensure
            update_available: Result.has_update
feature -- Status report
    computation_finished: BOOLEAN
        -- Does the blackboard contain the result of the computation?
feature -- Operation
    compute_result is
            -- Compute the result by selecting a specialist
        do
            from until computation_finished loop
                best_specialist.update_blackboard
            end
        end
end
```

An example implementation of the mathematical problem to solve equations on fractions described above using this framework can be downloaded at

http://se.inf.ethz.ch/teaching/ss2006/0050/exercises/blackboard_example.zip

4) Without the pattern

Without the pattern, it is necessary to explicitly schedule the algorithms provided through the specialists. Scheduling and algorithms will get mixed.

Possible advantages of such a solution:

- Easier to implement.
- Clearer understanding of the overall solution strategy.
- More efficient.

The disadvantages are:

- No separation of concerns (schedule vs. algorithms).
- More difficult to add new algorithms.
- More difficult to add new solution strategies.