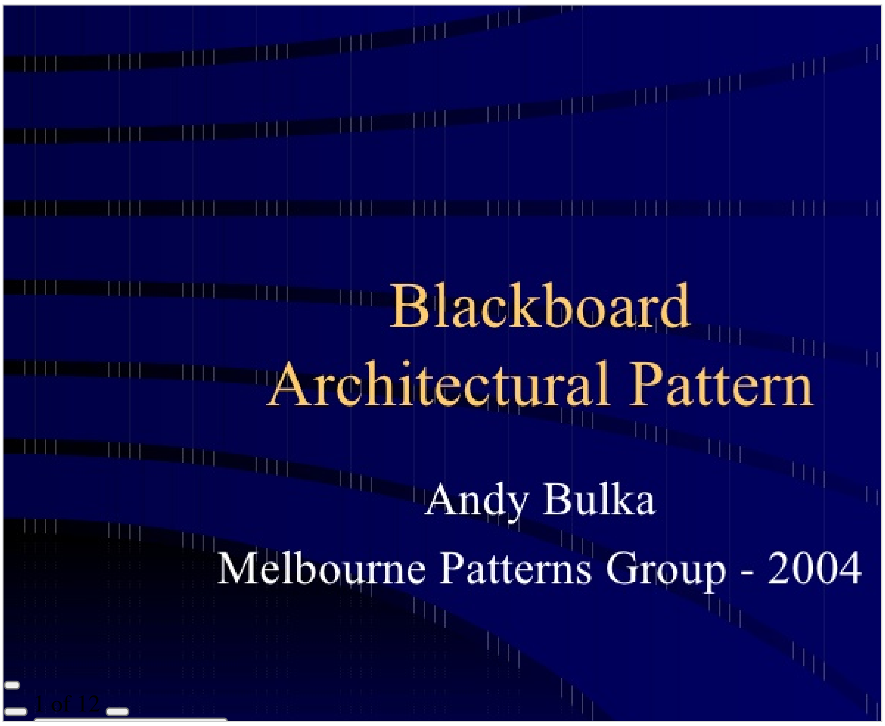
AndyPatterns

Blackboard Architectural Pattern

This pattern is useful for problems for which no deterministic solution strategies are known. In Blackboard several specialised sub-systems assemble their knowledge to build a possibly partial or approximate solution.

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



Blackboard
Architectural Pattern

Andy Bulka
Melbourne Patterns Group - 2004

1 of 12

Blackboard Pattern from **tcab22**

Example Code

This is a pattern that is difficult to do an example for because you would normally have to set up a large infrastructure of classes etc. I refused to be put off and presented a simple blackboard pattern implementation in python, during my talk to the Melbourne Patterns Group. Please forgive its limitations but it should at least give you an idea of what this pattern is about.

Note that you can flick a switch in the code below and have a small GUI pop up in Swing via jython.

```python
"""
  Blackboard system
  by Andy Bulka
  Prepared for the Melbourne Patterns group - August 2004
"""
import random
True = 1
False = 0

print "Welcome to my blackboard system"

class Blackboard:
    def __init__(self):
        self.experts = []
        self.commonState = {}

    def AddExpert(self, expert):
        self.experts.append(expert)

class Controller:
    def __init__(self, blackboard):
        self.blackboard = blackboard
        # init blackboard
        self.blackboard.commonState['answer'] = 0
        self.blackboard.commonState['answerCorrectness'] = 0

    def Loop(self):
        while self.blackboard.commonState['answerCorrectness'] < 90:
            candidates = []
            for expert in self.blackboard.experts:
                eagerness = expert.CanContribute()
                if eagerness:
                    candidates.append((eagerness,expert))  # append a tuple

            candidates.sort()    # move winning tuple to the end of the list
            winningExpert = candidates[-1][1]  # -1 means the last item in list.
            winningExpert.execAction()

        return self.blackboard.commonState['answer']

    def Loop_OLD(self):
        while self.blackboard.commonState['answerCorrectness'] < 90:
            for expert in self.blackboard.experts:
                if expert.CanContribute():
                    expert.execAction()
        return self.blackboard.commonState['answer']


class AbstractExpert:
    def __init__(self, blackboard, outtext):
        self.blackboard = blackboard
        self.outtext = outtext

    def CanContribute(self):
        raise 'not implemented'

    def execAction(self):
        raise 'not implemented'

class SmartAss(AbstractExpert):

    def CanContribute(self):
        return random.randint(1,20)

    def execAction(self):
        self.blackboard.commonState['answer'] += random.randint(1,20)
        print '.',
        self.outtext.text += '.'

class WiseMan(AbstractExpert):
    def CanContribute(self):
        if self.blackboard.commonState['answer'] > 200:
            return random.randint(1,20)
        else:
            return False

    def execAction(self):
        self.blackboard.commonState['answer'] += 1
        self.blackboard.commonState['answerCorrectness'] += 5
        print '*',
        self.outtext.text += '*'

WANT_JAVA_GUI = False

if WANT_JAVA_GUI:
    from javax.swing import JFrame, JLabel, JButton, JTextField

    class JHutton(JButton):
        pass

    class GUI:
        def __init__(self):
            f = JFrame()
            f.show()
            f.size = 200,200
            f.title = "Blackboard Jungle"

            f.contentPane.add(JLabel("Expert1"))

            self.txt1 = JTextField(30)
            f.contentPane.add(self.txt1)

            f.contentPane.add(JLabel("Expert2"))

            self.txt2 = JTextField(30)
            f.contentPane.add(self.txt2)

            button = JHutton("Think")
            f.contentPane.add(button)
            button.actionPerformed = self.onClick

            from java.awt import FlowLayout
            f.contentPane.layout = FlowLayout()

            f.pack()
            f.visible = 1
            self.f = f

            blackboard = Blackboard()
            blackboard.AddExpert( SmartAss(blackboard, self.txt1) )
            blackboard.AddExpert( WiseMan(blackboard, self.txt2) )
            self.c = Controller(blackboard)

        def onClick(self, event):
            result = self.c.Loop()
            print
            print result
            print 'done'

    gui = GUI()
else:
    # Pure text
    class DummyTextWidget:
        def __init__(self):
            self.text = ''

    import sys
    blackboard = Blackboard()
    blackboard.AddExpert( SmartAss(blackboard, DummyTextWidget()))
    blackboard.AddExpert( WiseMan(blackboard, DummyTextWidget()))
    c = Controller(blackboard)
    result = c.Loop()
    print
    print result
    print 'done'
```

Output

```
Welcome to my blackboard system
. . . . . . . . . . . . . . . . . . . * . * * * * * . * . * . * . . * * * . * . . * * . . * . . * . *
371
done
```

concrete5 - open source CMS © 2020 AndyPatterns.   All rights reserved. Sign In to Edit this Site