

Overview

- Structure-shy Object pattern
- Better separation of concerns using Structure-shy Traversal and Selective Visitor
- Cover now more details

10/17/00

AOO/Demeter

1

Structure-shy object pattern

- Parsing, class graphs etc.
- separation of concerns:
 - sentences : robust representation
 - objects : brittle representations
 - grammars: rules for constructing brittle rep. from robust rep.

10/17/00

AOO/Demeter

2

Parsing

- class graphs
- class dictionaries = class graphs + syntax
- legal objects
- printing function
- defines language defined by class dictionary
- ambiguous class graph: two distinct objects are mapped to same sentence

10/17/00

AOO/Demeter

3

Class graphs

- construction, alternation nodes and edges
- textual and graphical
- common parts, flat class graphs
- optional parts and repetition
- parameterized classes

10/17/00

AOO/Demeter

4

Parsing

- Is a class dictionary ambiguous?
- If not ambiguous, can define inverse of printing function: called parsing
- Goal: Want a fast parser. Want to quickly recognize an ambiguous class dictionary.

10/17/00

AOO/Demeter

5

Parsing

- Inherent problem: There is no algorithm to check for ambiguity. Shown by reduction: Could solve other undecidable problems.
- Invent condition which is stronger than non-ambiguity and which can be checked efficiently. Ok. to exclude some non-ambiguous grammars.
- LL(1) conditions.

10/17/00

AOO/Demeter

6

Parsing

- LL(1) condition 1: Alternatives of abstract class must have unique first sets.
- LL(1) condition 2: Deals with alternatives which may be empty. Requires follow sets.
- LL(1) conditions can be checked efficiently.
- Printing is a bijection between tree objects and sentences.

10/17/00

AOO/Demeter

7

Commercial parsing tools

- Yacc and Lex
- JavaCC (Java compiler compiler) LL(k) support
- JTree: define classes using a grammar and provides a universal visitor
- many more

10/17/00

AOO/Demeter

8

Context switch: Separation of concerns

- An introduction to aspect-oriented programming

10/17/00

AOO/Demeter

9

Better separation of concerns

- An old topic: Dijkstra 1976
- Avoid or minimize tangling

10/17/00

AOO/Demeter

10

Two types of concerns

- Generic concerns
 - structure
 - coordination
 - remote invocation
 - exception handling
 - efficiency
 - interoperability
 - migration
- Behaviors
 - sum
 - check
 - print
 - translate
 - Pricing
 - FrequentCustomerPricing

10/17/00

AOO/Demeter

11

```
interface ShapeI extends Remote {
    double get_x() throws RemoteException ;
    void set_x(int x) throws RemoteException ;
    double get_y() throws RemoteException ;
    void set_y(int y) throws RemoteException ;
    double get_width() throws RemoteException ;
    void set_width(int w) throws RemoteException ;
    double get_height() throws RemoteException ;
    void set_height(int h) throws RemoteException ;
    void adjustLocation() throws RemoteException ;
    void adjustDimensions () throws RemoteException ;
}

public class Shape
    implements ShapeI {
    protected AdjustableLocation loc;
    protected AdjustableDimension dim;
    public Shape() {
        loc = new AdjustableLocation(0, 0);
        dim = new AdjustableDimension(0, 0);
    }
    double get_x() throws RemoteException {
        return loc.x();
    }
    void set_x(int x) throws RemoteException {
        loc.set_x(x);
    }
    double get_y() throws RemoteException {
        return loc.y();
    }
    void set_y(int y) throws RemoteException {
        loc.set_y(y);
    }
    double get_width() throws RemoteException {
        return dim.width();
    }
    void set_width(int w) throws RemoteException {
        dim.set_w(w);
    }
    double get_height() throws RemoteException {
        return dim.height();
    }
    void set_height(int h) throws RemoteException {
        dim.set_h(h);
    }
    void adjustLocation() throws RemoteException {
        loc.adjust();
    }
    void adjustDimensions () throws RemoteException {
        dim.adjust();
    }
}

10/17/00

interface ShapeI extends Remote {
    double get_x() throws RemoteException ;
    void set_x(int x) throws RemoteException ;
    double get_y() throws RemoteException ;
    void set_y(int y) throws RemoteException ;
    double get_width() throws RemoteException ;
    void set_width(int w) throws RemoteException ;
    double get_height() throws RemoteException ;
    void set_height(int h) throws RemoteException ;
    void adjustLocation() throws RemoteException ;
    void adjustDimensions () throws RemoteException ;
}

class AdjustableLocation {
    protected double x, y;
    public AdjustableLocation(double x, double y) {
        x = x; y = y;
    }
    synchronized double get_x() { return x; }
    synchronized void set_x(int x) { x = x; }
    synchronized double get_y() { return y; }
    synchronized void set_y(int y) { y = y; }
    synchronized void adjust() {
        x = longCalculations();
        y = longCalculations();
    }
}

class AdjustableDimension {
    protected double width=0.0, height=0.0;
    public AdjustableDimension(double h, double w) {
        height = h; width = w;
    }
    synchronized double get_width() { return width; }
    synchronized void set_w(int w) { width = w; }
    synchronized double get_height() { return height; }
    synchronized void set_h(int h) { height = h; }
    synchronized void adjust() {
        width = longCalculations();
        height = longCalculations();
    }
}

10/17/00
```

AOO/Demeter

12

thread synchronization
remote interaction

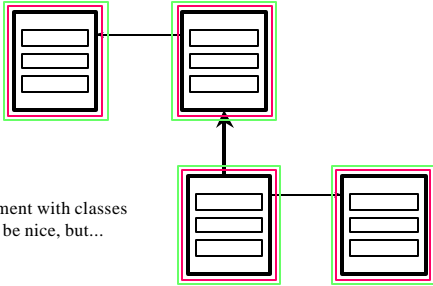
```
public class Shape
    implements ShapeI {
    protected AdjustableLocation loc;
    protected AdjustableDimension dim;
    public Shape() {
        loc = new AdjustableLocation(0, 0);
        dim = new AdjustableDimension(0, 0);
    }
    double get_x() throws RemoteException {
        return loc.getX();
    }
    void set_x(int x) throws RemoteException {
        loc.set_x(x);
    }
    double get_y() throws RemoteException {
        return loc.getY();
    }
    void set_y(int y) throws RemoteException {
        loc.set_y(y);
    }
    double get_width() throws RemoteException {
        return dim.getWidth();
    }
    void set_width(int w) throws RemoteException {
        dim.set_width(w);
    }
    double get_height() throws RemoteException {
        return dim.getHeight();
    }
    void set_height(int h) throws RemoteException {
        dim.set_height(h);
    }
    void adjustLocation() throws RemoteException {
        loc.adjust();
    }
    void adjustDimensions() throws RemoteException {
        dim.adjust();
    }
}
10/17/00 AOO/Demeter 13
```

```
interface ShapeI extends Remote {
    double get_x() throws RemoteException;
    void set_x(int x) throws RemoteException;
    double get_y() throws RemoteException;
    void set_y(int y) throws RemoteException;
    void set_width(int w) throws RemoteException;
    double get_height() throws RemoteException;
    void set_height(int h) throws RemoteException;
    void adjustLocation() throws RemoteException;
}

class AdjustableLocation {
    protected double x, y;
    public AdjustableLocation(double x, double y) {
        this.x = x; this.y = y;
    }
    synchronized double get_x() { return x; }
    synchronized void set_x(int x) { this.x = x; }
    synchronized double get_y() { return y; }
    synchronized void set_y(int y) { this.y = y; }
    synchronized void adjust() {
        x = longCalculation1();
        y = longCalculation2();
    }
}

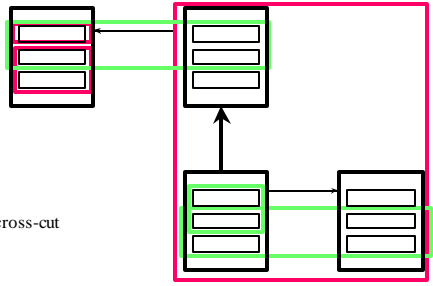
class AdjustableDimension {
    protected double width=0.0, height=0.0;
    public AdjustableDimension(double h, double w) {
        height_ = h; width_ = w;
    }
    synchronized double get_width() { return width_; }
    synchronized void set_w(int w) { width_ = w; }
    synchronized double get_height() { return height_; }
    synchronized void set_h(int h) { height_ = h; }
    synchronized void adjust() {
        width_ = longCalculation3();
        height_ = longCalculation4();
    }
}
10/17/00 AOO/Demeter 13
```

The source of tangling



Alignment with classes
would be nice, but...

The source of tangling



...issues cross-cut
classes

```
interface ShapeI extends Remote {
    double get_x() throws RemoteException;
    void set_x(int x) throws RemoteException;
    double get_y() throws RemoteException;
    void set_y(int y) throws RemoteException;
    void set_width(int w) throws RemoteException;
    double get_height() throws RemoteException;
    void set_height(int h) throws RemoteException;
    void adjustLocation() throws RemoteException;
}

class AdjustableLocation {
    protected double x, y;
    public AdjustableLocation(double x, double y) {
        this.x = x; this.y = y;
    }
    synchronized double get_x() { return x; }
    synchronized void set_x(int x) { this.x = x; }
    synchronized double get_y() { return y; }
    synchronized void set_y(int y) { this.y = y; }
    synchronized void adjust() {
        x = longCalculation1();
        y = longCalculation2();
    }
}

class AdjustableDimension {
    protected double width=0.0, height=0.0;
    public AdjustableDimension(double h, double w) {
        height_ = h; width_ = w;
    }
    synchronized double get_width() { return width_; }
    synchronized void set_w(int w) { width_ = w; }
    synchronized double get_height() { return height_; }
    synchronized void set_h(int h) { height_ = h; }
    synchronized void adjust() {
        width_ = longCalculation3();
        height_ = longCalculation4();
    }
}

public class Shape {
    protected AdjustableLocation loc;
    protected AdjustableDimension dim;
    public Shape() {
        loc = new AdjustableLocation(0, 0);
        dim = new AdjustableDimension(0, 0);
    }
    double get_x() throws RemoteException {
        return loc.getX();
    }
    void set_x(int x) throws RemoteException {
        loc.set_x(x);
    }
    double get_y() throws RemoteException {
        return loc.getY();
    }
    void set_y(int y) throws RemoteException {
        loc.set_y(y);
    }
    double get_width() throws RemoteException {
        return dim.getWidth();
    }
    void set_width(int w) throws RemoteException {
        dim.set_width(w);
    }
    double get_height() throws RemoteException {
        return dim.getHeight();
    }
    void set_height(int h) throws RemoteException {
        dim.set_height(h);
    }
    void adjustLocation() throws RemoteException {
        loc.adjust();
    }
    void adjustDimensions() throws RemoteException {
        dim.adjust();
    }
}
10/17/00 AOO/Demeter 16
```

Write this

Instead of
writing this

Code tangling is bad

- Harms program structure
- Distracts from main functionality
- Hard to program, error-prone
- Code difficult to understand, maintain

Ways to decrease the tangling

- Style guidelines
- Design patterns
- Better programming languages



Tangling of traversal/visitor calls

class graph

```
A = B.  
B = C.  
C = D.  
D = .
```

traversal + visitor calls

```
class A  
void t(V v){  
  v.before_b(this);  
  b.t(v);  
  v.after_b(this);  
}  
class B  
void t(V v){  
  v.before_c(this);  
  c.t(v);  
  v.after_c(this);  
}
```

visitor class V

```
before_b(A host) {  
  ... }  
after_b(A host) {  
  ... }  
...
```

10/17/00

AOO/Demeter

19

Tangling of behaviors V and W

class graph

```
A = B.  
B = C.  
C = D.  
D = .
```

```
visitor class V  
visitor class W
```

traversal + visitor calls

```
class A:  
void f() {t(new V(),  
  new W());}  
void t(V v,W w){  
  v.before_b(this);  
  w.before_b(this);  
  b.t(v);  
  w.after_b(this);  
  v.after_b(this);  
}
```

```
class B:  
void t(V v,W w){  
  v.before_c(this);  
  w.before_c(this);  
  c.t(v);  
  w.after_c(this);  
  v.after_c(this);  
}
```

At design level: class A: void f() to D (V,W)

10/17/00

AOO/Demeter

20

AOP = control of tangling

- DemeterJ controls the following tangling problems
 - structure/behavior,
 - navigation/behavior,
 - behavior/behavior,
 - object construction/structure,
 - synchronization/behavior,
 - remote invocation/behavior.

10/17/00

AOO/Demeter

21

Context switch

10/17/00

AOO/Demeter

22

Developing an adaptive program

- each simple behavior is translated into a visitor; localizes concerns; visitors are self-contained
- compose visitors, order visitors
- use top-down approach for writing program: first write method using visitors before visitors are implemented

10/17/00

AOO/Demeter

23

Container capacity problem

- recursive containers
- each contains items and has capacity
- each item has a weight
- find all containers above capacity
- leave details of containers vague

10/17/00

AOO/Demeter

24

Container capacity problem

- need to sum: use SummingVisitor
- need to check: use CheckingVisitor
- want to do it in one traversal: need to store sum when container is entered to compute difference after the container: DifferenceVisitor

10/17/00

AOO/Demeter

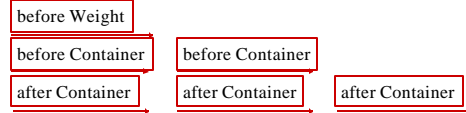
25

Traversal Summing Difference Checking
 before Weight before Container after Container after Container

Summary of code



Summary of event flow



Event based view of traversals and visitors

10/17/00

AOO/Demeter

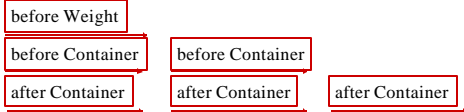
26

Traversal Summing Difference Checking
 before Weight before Container after Container after Container
 return s return d return v

Summary of code



Summary of event flow



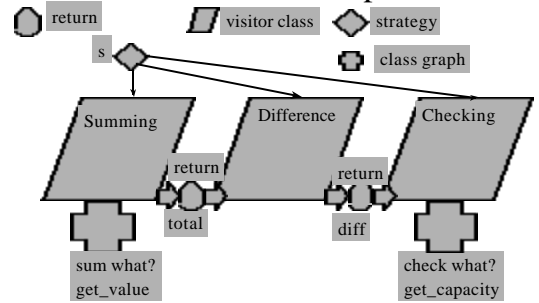
Event based view of traversals and visitors

10/17/00

AOO/Demeter

27

Visitor classes as components



10/17/00

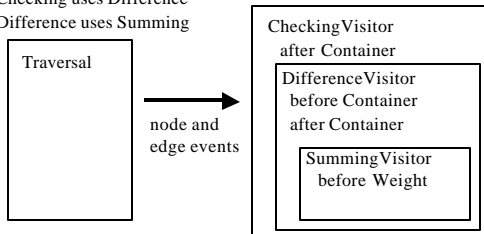
AOO

28

Karl Lieberherr:
com3360

Which Conceptual View? Most basic innermost

Checking uses Difference
 Difference uses Summing



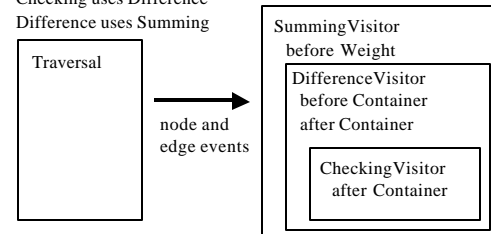
10/17/00

AOO/Demeter

29

Which Conceptual View? Most basic outermost: filter

Checking uses Difference
 Difference uses Summing



10/17/00

AOO/Demeter

30

Traversals are iterative

- from Container to Weight:
total = total + host.get_i();
- some recursive computations need extra attention using difference visitors

10/17/00

AOO/Demeter

31

Container capacity problem

- SummingVisitor
 - data member total, initialize to 0
 - add at Weight

10/17/00

AOO/Demeter

32

Container capacity problem

- CheckingVisitor
 - data member: violations
 - after each container, check whether difference of sum at the end and the beginning of a container (supplied by DifferenceVisitor) is larger than capacity
 - uses DifferenceVisitor
 - return violations

10/17/00

AOO/Demeter

33

Container capacity problem

- DifferenceVisitor
 - store sum at beginning (SummingVisitor) of container
 - return difference between sum at beginning of container and current sum upon request
 - uses SummingVisitor twice per container

10/17/00

AOO/Demeter

34

Container capacity problem

- Preliminary visitor class graph!
CheckingVisitor = <dV> DifferenceVisitor.
DifferenceVisitor = <sV> SummingVisitor.
- Visitor order: after container, both DifferenceVisitor and CheckingVisitor are active.
- DifferenceVisitor must be after CheckingVisitor (because after methods are in reverse order)

10/17/00

AOO/Demeter

35

Container capacity problem

```
traversal allWeights(CheckingVisitor cV, DifferenceVisitor dV,
    SummingVisitor sV) {to Weight;}
int checkCapacity() {
    CheckingVisitor cV = new CheckingVisitor();
    DifferenceVisitor dV = new DifferenceVisitor();
    SummingVisitor sV = new SummingVisitor();
    cV.set_dV(dV); dV.set_sV(sV); //link behaviors
    this.allWeights(cV, dV, sV); // call traversal with visitors
    return (cV.get_return_val()); }
```

cannot avoid all tangling, but can localize it. No longer spread over several classes.

10/17/00

AOO/Demeter

36

Container capacity problem

- Refined visitor class graph

```
CheckingVisitor =  
<dV> DifferenceVisitor <violations> int.  
DifferenceVisitor =  
<sV> SummingVisitor  
<difference> int  
<stack> Stack. // from java.util.*;  
SummingVisitor = <total> int.
```

10/17/00

AOO/Demeter

37

CheckingVisitor

```
init (@ violations = 0; @)  
after Container (@ diff = // use dV  
    cap = host.get_capacity()  
    if (diff > cap) violations = violations + 1; @)  
return int (@ violations @)
```

need to give return type since Demeter/Java
does not analyze code between (@ and @)

10/17/00

AOO/Demeter

38

DifferenceVisitor

```
init {{ stack = new Stack(); }}  
before Container {{ stack.push(... sV.get_total() ...); }}  
after Container {{ difference =  
    sV.get_return_val() - ... (Integer) stack.pop() ...; }}  
return int {{ difference }}  
//int get_current_sum() //traversing Visitor, flexibility  
    /// avoid accessing data members of non-primitive types  
    //to SummingVisitor // traversal strategy  
    // {before SummingVisitor // inlined visitor  
    // {{ return_val = host.get_total(); }} }
```

10/17/00

AOO/Demeter

39

SummingVisitor

```
init {{ total = 0; }}  
before Weight  
    {{ total = total + host.get_i(); }}  
return int {{ total }}
```

10/17/00

AOO/Demeter

40

Adaptiveness

- program we developed can be used with many different class graphs which satisfy a small set of constraints:
 - Container must contain Weight
 - Weight must have data member i of type int
 - Container must have data member capacity
- 2 examples

10/17/00

AOO/Demeter

41

Class graphs

```
Container = <contents> List(Item) <capacity> Capacity.  
Item : Container | Simple.  
List(S) ~ {S}.  
Simple = <name> Ident <w> Weight.  
Capacity = <i> int.  
Weight = <i> int.
```

alternative: move <w> Weight to Item

10/17/00

AOO/Demeter

42

Class graphs

Container : Container1 | Container2 *common*
<contents> List(Item) <capacity> Capacity.
Container1 = <packaging1> Simple.
Container2 = <packaging2> Simple.
Item : Container | Simple *common* <w> Weight.
List(S) ~ {S}.
Simple = <name> Ident.
Capacity = <i> int.
Weight = <i> int.