

Class Design and Evaluation

Design & Evaluation 1

- Class Design: Perspectives
- Example: Library System
- Behavioral Perspective
- Behavioral Categories
- Structural Perspective
- Structural Categories
- Informational Perspective
- Data Versus State
- Evaluating a Class Design
- Tests for Adequacy of Abstraction
- Good or Bad Abstractions?
- Tests for Adequacy of Responsibilities
- Tests for Adequacy of Interface
- Example of Poor Naming
- Tests for Adequacy of Usage
- Revised Location Class
- Implementation

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Class Design: Perspectives

Design & Evaluation 2

Behavioral
Emphasizes actions in system

Informational
Emphasizes role of information/data/state and how it's manipulated

Structural
Emphasizes relationships among components

specification

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Example: Library System

Design & Evaluation 3

Behavioral (actions):

- Patrons are registered
- Books are checked out

Structural (relationships):

- Catalog is made of books
- Book may be checked out to a patron

Informational (state):

- What's the status (available, checked out, ???) of a book?
- What books does a patron have checked out?

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Behavioral Perspective

Design & Evaluation 4

Consider some action in a program...

What object...

- initiates action?

What objects...

- help perform action?
- are changed by action?
- are interrogated during action?

Consider registering a patron...

Controller (procedural)...

- initiates the action

Circulation Desk...

- performs the action

Patron List...

- is changed by the action

Patron List...

- is interrogated during the action

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Behavioral Categories

Design & Evaluation 5

- Actor (does something)
Circulation Desk
- Reactor (system events, external & user events)
Controller, Parser??
- Agent (messenger, server, finder, communicator)
Catalog, PatronList
- Transformer (data formatter, data filter)
Parser

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Structural Perspective

Design & Evaluation 6

What objects...

- are involved in relationship?
- are necessary to sustain (implement, realize, maintain) relationship?

What objects not in relationship...

- are aware of and exploit relationship?

Consider a relationship: book is checked out to patron
Circulation Desk...

- is involved in the relationship
Catalog and PatronList...
- are necessary to sustain the relationship

???

- is aware of and exploits the relationship

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Structural Categories

Design & Evaluation 7

- Acquaintance (symmetric, asymmetric)
 - CirculationDesk knows about PatronList, asymmetric relationship
- Containment (collaborator, controller)
 - CirculationDesk controls/uses PatronList and Catalog
- Collection (peer, iterator, coordinator)
 - PatronList contains and manages Patrons
 - CirculationDesk contains and manages CheckedOut objects

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Informational Perspective

Design & Evaluation 8

What objects...

- represent the data or state?
- read data or interrogate state?
- write data or update state?

Consider a state: status of book

CheckedOut list and Catalog implicitly...

- represent (stores) the state information

CirculationDesk...

- interrogates the state of a book (via ...)

CirculationDesk...

- updates the state of a book

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Data Versus State Design & Evaluation 9

Data	State
<p><u>Definition:</u> Information processed by the system</p> <p><u>Example:</u> checkout command</p>	<p><u>Definition:</u> Information used by system to control processing</p> <p><u>Example:</u> BookStatus (Avail, CheckedOut, etc.)</p>

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Evaluating a Class Design Design & Evaluation 10

Evaluation is needed to accept, revise or reject a class design.

Five aspects to be evaluated:

- Abstraction: useful?
- Responsibilities: reasonable?
- Interface: clean, simple?
- Usage: “right” set of methods?
- Implementation: reasonable?

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Tests for Adequacy of Abstraction Design & Evaluation 11

Identity:
Are class purpose and method purposes well-defined and connected?

Clarity:
Can purpose of class be given in brief, dictionary-style definition?

Uniformity:
Do operations have uniform level of abstraction?

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Good or Bad Abstractions? Design & Evaluation 12

class Date:
Date represents a specific instant in time, with millisecond precision.

class TimeZone:
TimeZone represents a time zone offset, and also figures out daylight savings.

Computer Science Dept Va Tech February 2000 OO Software Design and Construction ©2000 McQuain WD

Tests for Adequacy of Responsibilities

Design & Evaluation 13

Clear:

Does class have specific responsibilities?

Limited:

Do responsibilities fit the abstraction (no more/less)?

Coherent:

Do responsibilities make sense as a whole?

Complete:

Does class completely capture abstraction?

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Tests for Adequacy of Interface

Design & Evaluation 14

Naming:

Do names clearly express the intended effect?

Symmetry:

Are names and effects of pairs of inverse operations clear?

Flexibility:

Are methods adequately overloaded?

Convenience:

Are default values used when possible?

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Example of Poor Naming

Design & Evaluation 15

```
class ItemList {
private:
// . . .
public:
void Delete(Item item);
// Take Item's node out of list and delete Item

void Remove(Item item);
// Take Item's node out of the list but do not
// delete Item

void Erase(Item item);
// Keep Item's node in List, but with no information
};
```

Hard to remember difference!

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

Tests for Adequacy of Usage

Design & Evaluation 16

Examine how objects of the class are used in different contexts (see below...)

Incorporate all operations that may be useful in these contexts... up to a point...

```
class Location {
private:
int xCoord, yCoord; //coordinates
public:
Location(int x, int y);
int xCoord(); //return xCoord value
int yCoord(); //return yCoord value
};

// usage:
Location point(100,100);
// shift point:
point = Location( point.xCoord()+5, point.yCoord()+10 );
```

It's so complex!

Computer Science Dept Va Tech February 2000

OO Software Design and Construction

©2000 McQuain WD

```
class Location {
private:
    int xCoord, yCoord; //coordinates
public:
    Location(int x, int y);
    int XCoord();      //return xCoord value
    int YCoord();      //return yCoord value
    void ShiftBy(int dx, int dy); // shift by relative coordinates
};

// Revised usage:
Location point(100,100);

point.ShiftBy(5, 10); // shift point
```

Least important, mostly easily changed aspect to be evaluated.

- poorly engineered design leads to problematic implementation
- massaging a problematic implementation (without redesign) rarely produces any effective improvement
- it's only code...

Overly complex implementation may mean:

- class is not well conceived
- class has been given too much responsibility